Enhancing the Decoding of Short LDPC Codes with Stochastic Sequences

Master's Thesis

By

Yuyuan Gu and Haien Li

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

LUND UNIVERSITY

2021

Abstract

Low-Density Parity-Check (LDPC) codes are one of the most popular codes used in nowadays' communication due to their high coding efficiency at low decoding complexity. With these characteristics, LDPC codes are suitable for high-speed information transmission systems. The widely used decoding algorithm for LDPC codes is Belief Propagation (BP) decoding with which the performance of LDPC codes can approach the Shannon limit.

With the development of 5G, a lot of attention is given to the so-called ultra-reliable low latency communication scenario which needs the short code to transmit. However, short codes will not have a very good performance with BP decoding. To solve this problem, investigations have been done such as multiple-bases BP decoding. Although this decoding method has a better performance than BP decoding, it will cause a high complexity in the hardware implementation. Besides, an investigation on the stochastic decoding for short codes was proposed in 2003. This decoding method is hardware friendly, but the performance of it can only approach BP decoding.

Inspired by multiple-bases BP decoding and stochastic decoding, binary stochastic decoding with parallel decoders is proposed in this thesis firstly. It represents stochastic sequences with multiple parallel Tanner graphs and uses hard-decision decoding in the iterative part because of the binary input bits. However, this decoding method has a severe performance loss compared to BP decoding. To avoid this problem, the enhancement method is used to make the binary sequences be non-binary sequences that can form more powerful parallel decoders. Then the non-binary symbols of these new sequences are transformed to their corresponding log-likelihood ratio which enables the iterative part to use BP decoding. In addition, combining with the ML decision of list decoding, the decision part of our stochastic decoder can fully utilize the output to increase the efficiency of the decoding method.

After ensuring the performance of our non-binary stochastic decoding to be better than BP decoding, the complexity of the decoder is reduced to save the computational resources. Finally, with constant adjustments, the bit width of each non-binary symbol is determined to be 15, and the sequence length is reduced to 20 which can let the non-binary stochastic decoding has an acceptable complexity while keeping good performance.

Acknowledgments

Firstly, we would like to appreciate our supervisor at Lund University, Prof. Michael Lentmaier, for his patient and helpful instruction. Michael always taught us like an elder and guided us when we feel frustrated and helpless during our thesis work. With his help, we were able to stay in the right direction and develop our ideas.

Then, we are grateful to Dr. Wei Zhou at Huawei Lund for his powerful help in the initial time of our simulation work and Dr. Guoda Tian at Lund University for helping us solve the other things of the university when we are busy at doing the thesis.

Lastly, we are grateful to our family and friends for their constant companion and support.

Yuyuan Gu

Haien Li

Lund, May 2021

Contents

List of Figures

List of Tables

List of Acronyms

ECC	Error Correction Code
LDPC	Low Density Parity Check
URLLC	Ultra-reliable Low Latency Communication
VN	Variable Node
CN	Check Node
BP	Belief Propagation
BSC	Binary Symmetric Channel
AWGN	Additional White Gaussian Noise
MRF	Markov Random Field
LLR	Log-Likelihood Ratio
ML	Maximum Likelihood
BPSK	Binary Phase Shift Keying
SNR	Signal-to-Noise Ratio
BER	Bit Error Rate
FER	Frame Error Rate
QC	Quasi Cyclic

1. Introduction

1.1. Background

In the process of transmitting digital signals, errors often occur in the transmitted data stream which will cause bad signals such as blurred images, discontinuous sound, and so on. Channel coding which can be divided into encoding and decoding is an important step in the whole communication system to protect the digital signals and correct the errors. In the encoding part, error correction codes (ECC) are used to encode digital signals. In the decoding part, the received signal can be corrected by corresponding decoding algorithms. Typically, the transmitted vector is $\mathbf{u}$, the encoded vector is $\mathbf{v}$, the received vector is $\mathbf{r}$, and the decoded codeword is $\hat{\mathbf{v}}$. The simplified process of the basic transmission in AWGN (Additional White Gaussian Noise) channel is shown in the Fig. 1.1.

Fig. 1.1 Basic transmission in AWGN channel

An ECC called LDPC code was first proposed by Gallager in his doctoral dissertation in the 1960s [1]. But limited to the technical conditions at the time and lack of feasible decoding algorithms, LDPC codes were ignored in the following 35 years. In the meantime, Tanner popularized the LDPC code in 1981 and gave a graph representation of the LDPC codes [2], which was later called the Tanner graph. The Tanner graph also lays the foundation for LDPC decoding, which will be introduced in detail in the second part. Around 1995, MacKay and Neal et al. [3] re-researched LDPC codes and proposed feasible decoding algorithms, which further discovered that LDPC codes have good performance close to the Shannon limit.

1.2. Motivation

In the context of 5G and beyond, a lot of attention is given to the so-called ultra-reliable low latency communication (URLLC) scenario, which is relevant for applications like smart transportation, factory automation, or robotic surgeries. In such a scenario the latency is a major concern, and code should not be very long. For short code, the traditional BP decoding is not the best option, because its performance will decrease as the code length gets

smaller [4]. Using maximum likelihood (ML) decoding with the LDPC code could avoid this problem. But the complexity of the ML algorithm will increase exponentially with the code length and become infeasible in practice. Thus, how to keep the low latency and high reliability at the same time is a major open problem with short LDPC codes.

Based on the traditional algorithms used in the LDPC decoding, some research has already been done to tackle the decoding problem of short codes. Hehn et al. proposed a method called multiple-bases BP decoding in 2007 [5], which uses several graphs of the same code and decodes the code with the BP algorithm parallelly. This method can enhance the decoder against the impact of short cycles in the Tanner graph, but it causes the high complexity on the hardware implementation. And in 2019, Zhou et al. [6] applied some list decoding techniques to short 5G LDPC codes to reduce the gap to the ML decoding performance. Rapley et.al proposed the concept of stochastic decoding in 2003 [7], which can meet the performance of a BP decoder for the (7,4) Hamming code. Although this method will not cause a high physical complexity, it cannot beat the BP decoding.

1.3. Contributions

This master's thesis work aims to investigate a specific solution to improve the performance of the stochastic decoder when decoding a short block code. Inspired by [5] and [7], firstly we will represent the inputs of different parallel graphs with binary stochastic sequences which can be seen as parallel decoders. However, the binary stochastic decoder cannot beat the BP decoding. Then, due to the weakness of the binary stochastic decoder, more powerful parallel decoders with non-binary sequences are proposed. Lastly, inspired by [6], we use average decision, hard ML decision and soft ML decision to deal with the outputs to see how we can increase the efficiency of our decoder. After comparing the results of these three decision methods, the soft ML decision is chosen to be the final decision method of our non-binary stochastic decoder. With non-binary sequences and soft ML decision, the performance of stochastic decoder is improved to be better than the performance of BP decoding. Next, in order to save the computational resources used by stochastic decoder, the complexity will be reduced while keeping the decoder a good performance.

During the thesis work, the implementation in MATLAB and simulation study is carried out by Yuyuan and Haien together. About the writing of the thesis report, Yuyuan is in charge of Section 1-3 and Haien takes the response for Section 4-6. Section 7-8 are finished by Yuyuan and Haien together.

1.4. Structure of the Thesis

In this thesis, Section 1 is about the brief background and structure of the thesis.

Section 2 introduces the main knowledge of LDPC and its classic decoding algorithms. Besides these, Section 2 also introduces the basic stochastic decoding method to make a start for our decoding methods.

Section 3 is mainly about our parallel decoders with hard-decision decoding and the regeneration method which can improve the performance of our binary stochastic decoding method.

The combination of list decoding and binary stochastic decoding is introduced in Section 4. In Section 4, we also compare the performance of stochastic list decoding with hard input and the stochastic list decoding with soft input in order to state the inspiration of our non-binary stochastic decoding method.

Then, Section 5 starts with the generation of the non-binary stochastic sequence to introduce our non-binary stochastic decoding method, and then adjust the parameters to reduce the decoding complexity while keeping a good performance.

Section 6 shows the final result of our proposed decoding methods and the application of stochastic decoding with non-binary input sequences to other LDPC codes.

Then, Section 7 gives a brief summary of our thesis work. Finally, some future work is proposed in Section 8.

2. Background on LDPC Codes

2.1. Structure of LDPC Codes

LDPC stands for Low-Density Parity-Check. Low-density means that an LDPC code is a linear block code with a sparse parity-check matrix [1]. Due to the sparseness of parity-check matrix, its decoding complexity and code length are not in an exponential relationship, but in a linear relationship. Therefore, the code length of the LDPC code can be very long, reaching thousands to tens of thousands or even higher. One of the advantages brought by this is that the correlation length between the bits in a codeword is relatively long, and the correlation of the bits in the codeword is fully utilized in the decoding part to improve the decoding accuracy, and it also makes full use of the channel feature [8].

LDPC codes include regular LDPC codes and irregular LDPC codes. In a regular LDPC code, each column of the **H** matrix has the same number of ones, otherwise it is called an irregular LDPC code [1]. Suppose in a **H** matrix with size of $N-K \times N$, the number of ones in each column is d_v, the number of ones in each row is d_c. Then there should be $d_v \ll N-K$ and $d_c \ll N$, while d_v represents column weight and d_c represents row weight. Therefore, LDPC codes are usually represented by the above three parameters (N, d_v, d_c). Fig. 2.1 shows the example of the **H** matrix of (9,2,3) LDPC.

In this work, we choose a (1024,3,6) LDPC code as long code and a (126,3,6) LDPC code as short code. The d_c of these check matrices are both 6, and the d_v are both 3.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 2.1 Parity-check matrix of (9,2,3) LDPC code

In addition, in the **H** matrix, the condition that values of the same column in any two adjacent rows all equal to 1 cannot happen more than one time. That is, there should be no rectangle whose four corners are constructed by four ones in the **H** matrix. And such rectangle is called 4-cycle. Typically, the 4-cycle will limit the performance of the LDPC code and cause the bit error

rate of the decoder not to converge to the correct solution [9]. Fig. 2.2 shows an example of a 4-cycle.

$$\begin{bmatrix} & \vdots & \vdots & \\ \cdots & 1 & 1 & \cdots \\ \cdots & 1 & 1 & \cdots \\ & \vdots & \vdots & \end{bmatrix}$$

Fig. 2.2 An example of a 4-cycle in the **H** matrix

2.2. Tanner Graph

The Tanner graph represents the parity-check matrix of an LDPC code [2]. It contains two types of nodes, variable nodes (VN) corresponding to each column of the parity-check matrix and check node (CN) corresponding to each row of the parity-check matrix. Each row of the parity-check matrix represents a parity-check equation, and each column represents a codeword bit. If a codeword bit is included in the corresponding check equation, then a connection is used to connect the involved VN and the CN, so the number of connections in the Tanner graph is equal to the number of ones in the check matrix. In the typical Tanner graph, VNs are represented by circles, and CNs are represented by squares [10]. Fig. 2.4 shows the Tanner graph of the (9,2,3) LDPC code from Fig. 2.3.

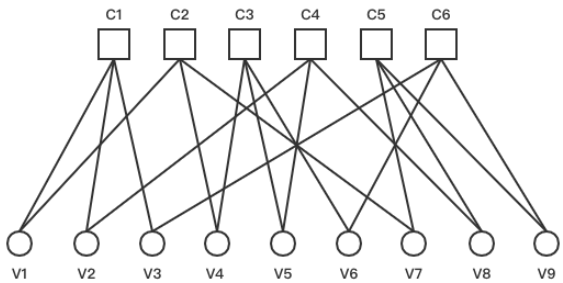


Fig. 2.3 Tanner graph of a (9,2,3) LDPC code

In Fig. 2.3, C1-C6 represent six CNs, that is, the 6 rows of the parity-check matrix **H** from top to bottom. V1-V9 represent 9 VNs, that is the 9 columns of the parity-check matrix **H** from left to right. In the **H** matrix, there are three ones in each row, which means that each CN is connected to three

VNs by some edges, so the degree of each CN is 3. In the same way, each VN will be connected to two CNs by some edges, because each column in the **H** matrix has 2 ones, and the degree of each VN is 2. Since $\mathbf{v} \cdot \mathbf{H}^{\mathrm{T}} = 0$, The relationship between VNs and CNs can also be represented as

$$\begin{aligned} V_1 \oplus V_2 \oplus V_3 &= 0, \\ V_1 \oplus V_4 \oplus V_7 &= 0, \\ V_4 \oplus V_5 \oplus V_6 &= 0, \\ V_2 \oplus V_5 \oplus V_8 &= 0, \\ V_7 \oplus V_8 \oplus V_9 &= 0, \\ V_3 \oplus V_6 \oplus V_9 &= 0. \end{aligned} \tag{1}$$

A cycle in the Tanner graph is composed of a group of nodes connected to each other in the graph. The cycle takes one of the group of nodes as the start and end at the same time. And it will pass through each node only once. The length of the cycle is defined as the number of lines it contains. The girth of the Tanner graph is defined as the smallest cycle length in the graph [10]. Continuing to use the Tanner graph of the (9, 2, 3) LDPC code as an example, we can see that the red line in Fig. 2.4 represents a cycle, the girth of this cycle is 8. In this Tanner graph, since there are no any other cycles whose length is smaller than 8, the girth of this Tanner graph is 8.

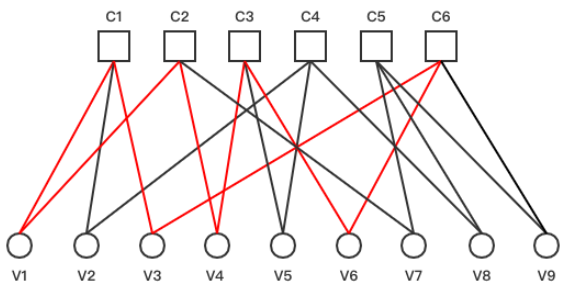


Fig. 2.4 Girth of Tanner graph of a (9,2,3) LDPC code

The classical iterative decoding is based on the Tanner graph, and the posterior probability message or the hard-decision result is transmitted between VNs and CNs [11]. This information will be transmitted again to the connected nodes after being updated by the algorithm. The condition for exiting iterative decoding is to obtain the valid codeword which is determined

by $\hat{\mathbf{v}} \cdot \mathbf{H}^{\mathrm{T}} = 0$ or reach the maximum number of iterations, i_{max}. This stopping rule will be used in the whole thesis work.

2.3. Decoding Algorithms

2.3.1. Belief Propagation Decoding

Belief propagation (BP), also known as sum-product algorithm, is a message passing algorithm that performs inference on a graph model and can be used in Bayesian networks and Markov random domains (e.g. AWGN channel). The belief propagation algorithm updates the current marking state of the entire Markov Random Field (MRF) by using the mutual information between CNs and VNs. It is an approximate calculation based on MRF. The algorithm is an iterative method that can solve the problem of probabilistic inference in the probabilistic graph model, and the dissemination of all information can be realized in parallel. After multiple iterations, the reliability of all nodes no longer changes. At this time, the mark of each node is said to be the optimal mark, and the MRF has also reached a state of convergence. For MRF without cycles the BP algorithm can converge to its optimal solution [12].

For LDPC codes, BP decoding is a soft decoding method. The Tanner graph is used as the transmission network, and the Log-Likelihood Ratio (LLR) is transmitted between the VN and the CN. The initial LLR from the channel, L_{ch}, can be obtained by [13]

$$L_{ch} = \ln \frac{P(r|v=0)}{P(r|v=1)} \ . \tag{2}$$

In BP decoding, the update of the CN only needs to consider the information provided by the connected edges (VNs). The update of the VN must not only consider the information provided by the connected edges (CNs), but also consider the initial LLR, L_{ch} [13].

Fig. 2.5 shows the information delivery process when updating the CN. Here, $L_v(e_1)$ and $L_v(e_2)$ represent the information passed from the VN to the CN. $L_c(e_3)$ indicates that the CN will update the information and pass it to the target VN through the red line.

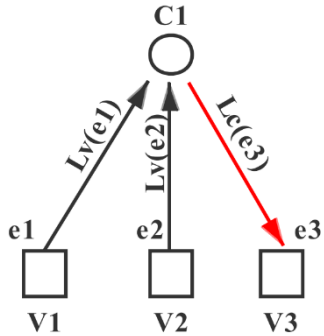


Fig. 2.5 Information delivery process of updating CNs

Fig. 2.6 shows the message passing process when updating VNs. This process is similar to that of updating VNs. $L_c(e_1)$ and $L_c(e_2)$ represent the information passed from the CN to the VN, and L_{ch} represents the initial channel output value. $L_v(e_3)$ indicates that the updated information of the VN will be passed to the target CN through the red line.

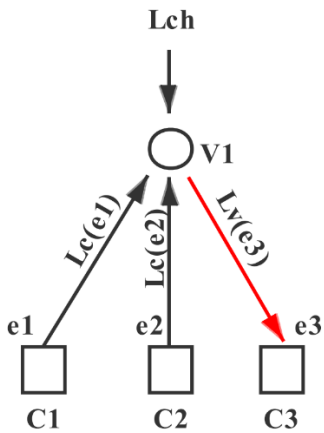


Fig. 2.6 Information delivery process of updating VNs

The specific BP decoding process [14] is as follows:

Step 1: Initial input is L_{ch}. Setting i=1, where i is the counter of iteration. And giving the initial input as the initial updated value to all of edges of the VNs, as shown in (3).

$$L_v^{(0)}(e_j) = L_{ch}(v) \tag{3}$$

Step 2: Equation (4) is used to update the CNs.

$$L_c^{(i)}(e_k) = 2\tanh^{-1}\left(\prod_{k\prime \neq k} \tanh\left(\frac{1}{2}L_v^{(i-1)}(e_{k\prime})\right)\right) \tag{4}$$

Step 3: Equation (5) is used to update the VNs.

$$L_v^{(i)}(e_j) = L_{ch}(v) + \sum_{j\prime \neq j} L_c^{(i)}(e_{j\prime}) \tag{5}$$

Step 4: If the number of iterations reaches the maximum value, i_{max}, the function that be used to calculate the output value of the current VN is shown in (6).

$$L_{out} = L_{ch}(v) + \sum_j L_c^{(i_{max})}(e_j) \tag{6}$$

2.3.2. Hard-decision Decoding

The basic principle of BP decoding can also be applied to the Binary Symmetric Channel (BSC). We call this decoding method iterative hard-decision decoding. In hard-decision decoding, the information transmitted between the VN and the CN is no longer Log-Likelihood, but 0/1 bit. The specific decoding steps [16] are as follows.

Step 1: First, make a hard-decision based on the Log-Likelihood (L_{ch}) which is the output of the channel. When L_{ch} is greater than 0, the judgment result is 0. On the contrary, the judgment result is 1. From this, we can get an initial sequence of 0/1, $\mu_{ch}(v)$. This sequence is the initial input that will be brought into the decoder. Thus, we can set *i*=1, where *i* is the counter of iterations. And giving the initial input as the initial updated value to all of edges of the VNs as (7) shows.

$$\mu_v^{(0)}(e_k) = \mu_{ch}(v) \tag{7}$$

Step 2: In order to satisfy all of the check equations. Equation (8) is used to update the CN.

$$\mu_c^{(i)}(e_k) = \sum_{k\prime \neq k} \mu_v^{(i-1)}(e_{k\prime}) \mod 2 \tag{8}$$

Step 3: In the update of the VN, we use the majority rule to update the edge and final output as (9) shows.

$$\mu_v^{(i)}(e_j) = \begin{cases} \bar{\mu}_{ch}(v) & if \ \left|\left\{j' : \mu_c^{(i)} \neq \mu_{ch}(v), j' \neq j\right\}\right| \geq \frac{d_v - 1}{2} \\ \mu_{ch}(v) & else \end{cases} \tag{9}$$

Among them, d_v represents the degree of each VN as mentioned in Section 2.1, that is, the number of edges connected to each VN. This formula indicates that when the target edge is updated, the number of the remaining edges which provide different bit from the original input is calculated. If the counted number is greater than half of the degree of each VN, the initial value after the flip is assigned to this edge, that is, 0 becomes 1 or 1 becomes 0. In the same way, when updating the final output, we should count how many edges provide different information from the original input among all the edges. If the counted number is greater than half of the degree of each VN, the initial value after the flip is assigned to the final output, otherwise, the initial value is directly assigned to the final output.

2.3.3. Stochastic Decoding

The stochastic decoding is first proposed in [7]. This section only reviews the concept of the basic stochastic method which is the inspiration of our decoding method.

The idea of basic stochastic decoding is based on the stochastic computation [7] that it converts the L_{ch} obtained from the channel into a probability value, and then generates a stochastic sequence corresponding to each message based on its probability of 0/1. The detailed steps are as follows:

Step 1: Based on (2), probability of 0/1 can be obtained from L_{ch}. Equation (10) shows the calculation function of the probability.

$$\begin{cases} P_0 = \dfrac{e^{L_{ch}}}{1 + e^{L_{ch}}} \\ P_1 = 1 - P_0 = \dfrac{1}{1 + e^{L_{ch}}} \end{cases} \tag{10}$$

Step 2: Based on (10), the stochastic sequence of corresponding probability can be generated. Here, we assume the calculated probability of 1 is 0.2 and length of sequence is 20. Then, the corresponding example sequence is shown in Fig. 2.7.

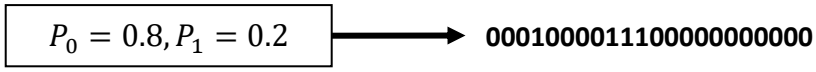


Fig. 2.7 A possible sequence for the probability of 0.2

In Fig. 2.8, it can be clearly seen that the number of 0 and 1 in this possible sequence meets with the channel output probability of 0.8 and 0.2.

In [7], the definition of the basic stochastic VN has been found. Taking $d_v = 3$ as an example, each VN is connected to three edges (CNs). Except the destination sequence which need to be updated, the update of the VN is completed by the participation of two stochastic sequences and the initial input sequence. We can use the probability of 1 in each sequence, that is $P_A = \Pr(e_1 = 1)$, $P_B = \Pr(e_2 = 1)$ and $P_{ch} = \Pr(S_{ch} = 1)$ to represent the three input stochastic sequences probability. According to the BP algorithm, the output probability P_Y $(e_3\_update = 1)$ is calculated by

$$P_Y = \frac{P_A P_B P_{ch}}{P_A P_B P_{ch} + (1-P_A)(1-P_B)(1-P_{ch})} . \tag{11}$$

Fig. 2.8 shows the hardware structure of the basic stochastic VN to perform (11).

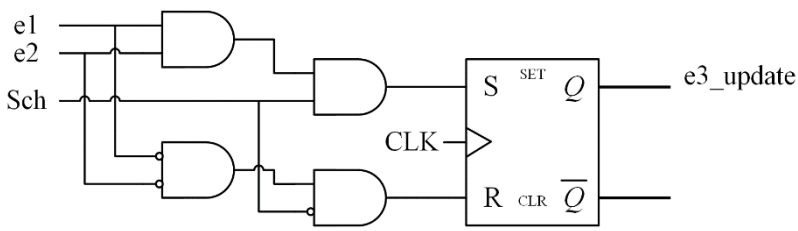


Fig. 2.8 Hardware structure of the basic stochastic VN

In [7], the definition of the basic stochastic CN has also been found. Taking $d_c = 3$ as an example, each CN is connected to three edges (VNs). Except the destination sequence which need to be updated, the update of the CN is completed by the participation of two stochastic sequence. Like the definition of the basic VN, the probability of occurrence of 1 in each sequence can still be used to represent the input stochastic sequences probability. According to the BP algorithm, the output probability P_Y $(e_3\_update = 1)$ is calculated by

$$P_Y = P_A(1 - P_B) + P_B(1 - P_A) . \tag{12}$$

Fig. 2.9 is the hardware structure of the basic stochastic CN to perform (12). From this hardware structure, we can clearly see that the update of the CN is actually the XOR operation.

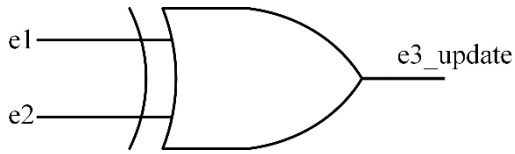


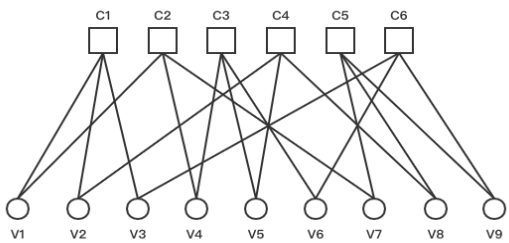
Fig. 2.9 Hardware structure of the basic stochastic CN

3. Binary Stochastic Decoding with Regeneration

3.1. Stochastic Decoding with Parallel Decoders

The stochastic decoding with parallel decoders is proposed based on the hard-decision decoding and stochastic computation. This method which is hardware friendly has simpler circuit than multiple-bases BP decoding. But unlike the stochastic decoding in Section 2.3.3, the updated method of the variable node is majority decision instead of Fig. 2.8 in our decoding method.

In the work of this section, the first step is that the L_{ch} corresponding to each message is converted to the stochastic sequence as described in the stochastic computation. And then, we will have an input table. The example of the input table is shown in Fig. 3.1. The number of rows of this table is the length of sequence L and the number of columns of this table is the number of VNs. Each row of this table can be seen as a candidate input, and all of the rows can be seen as the parallel decoders.



V1	V2	V3	V4	V5	V6	V7	V8	V9
0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1	0
0	1	0	0	1	0	1	1	0
0	0	1	0	0	0	1	0	0
0	0	1	0	0	1	1	0	0
1	1	1	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0

Fig. 3.1 An example of the input table of (9,2,3) LDPC code

These inputs will be taken into the iterative decoding separately. In the part of iterative decoding, the update of the CN and VN is the same as (8) and (9) of hard-decision decoding. When the iterative decoding ends, we will get an output table. Then, the majority rule can be used on each column of this table to get the final output of each VN. However, this stochastic decoding method has a severe performance loss whether it is used to decode long LDPC codes or short LDPC codes. Fig. 3.2 and Fig. 3.3, respectively, show the

decoding performance of our stochastic decoding when applied to a (126,3,6) regular LDPC code and a (1024,3,6) LDPC code. It should be noted that i_{max} used in all simulations in this thesis work is 64 which is an empirical value from [17].

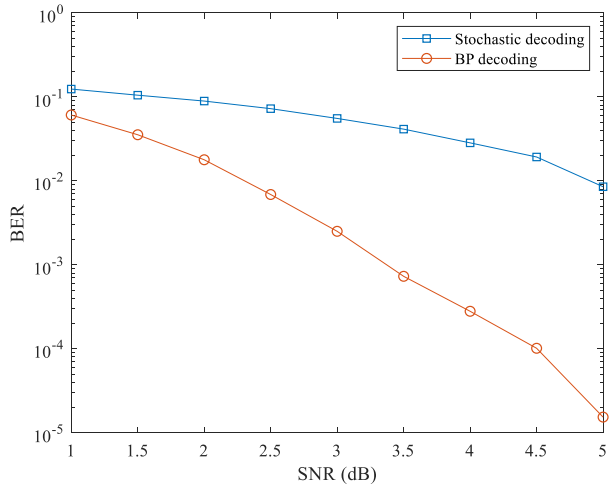


Fig. 3.2 Comparison of stochastic decoding and BP decoding, (126,3,6) LDPC code, L=1000

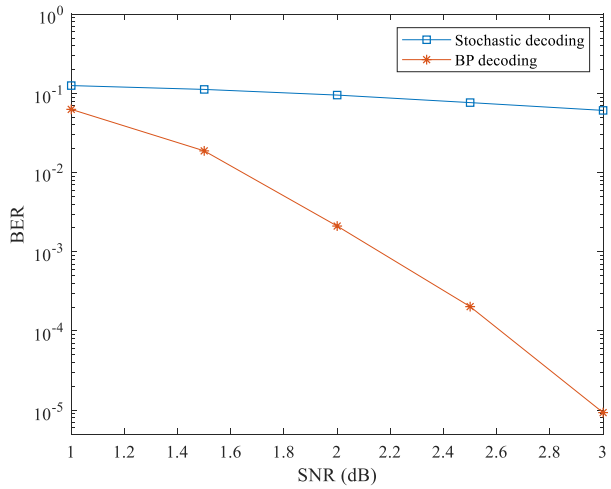


Fig. 3.3 Comparison of stochastic decoding and BP decoding, (1024,3,6) LDPC code, L=1000

3.2. Proposal of the Regeneration Method

As mentioned above, the stochastic decoding in Section 3.2 has a severe performance loss. After making a research on [7], the reason why the stochastic decoding with parallel decoders has the bad performance has been found.

According to the definition of the basic stochastic VN in [7], It can be seen from (11) that during the update process, the probability of the output sequence in the stochastic decoding should be equal to the output probability obtained in the BP decoding. In other words, in the update of VNs, we should pay more attention to the probability distribution of 0 and 1 of the entire output sequences. In the initial step of the iterative decoding process of Section 3.1, the input value of each VN has only one bit, so each update of the output value can only get one bit. However, we cannot judge whether the current updated output value satisfies the output probability obtained in BP decoding based on only one bit. Therefore, the whole input sequences of each message should be taken to participate in the iterative decoding part of stochastic decoding, so that the updated output value will also be a whole sequence.

And in the hardware structure which is shown in Fig. 8, it should be noted that a JK flip-flop is used in the hardware structure. Thus, only when $e_1 = e_2 = S_{ch}$, the output state of e_3 can be regarded as a regular state. Otherwise, the output value will continue to use the output value of the previous iteration, and we call this state the hold state. Typically, the result of regular state is more reliable than the result of the hold state. The state diagram of the basic VN is shown in Table 1.

Table 1. State diagram of the basic stochastic VN

e_1	e_2	S_{ch}	e_3
0	0	0	0 (regular state)
1	1	1	1 (regular state)
0	1	0	hold state
1	0	1	hold state
1	1	0	hold state
0	1	1	hold state

Inspired by the concept of this hardware structure, we can find that only the results judged by 000 and 111 are the most reliable. But in the parallel decoding algorithm, it only uses the majority decision when updating VNs. When we focus on the probability of 0 judged by 000 and probability of 1 judged by 111 in the output sequence generated by the majority decision, the actual probability of such 0 (regular state) and actual probability of such 1 (regular state) cannot add up to 1.

To revise the actual probability of 0/1 in the output sequence, we need to calculate the number of 0/1 judged by 000/111 to recalculate P_0 and P_1, and regenerate the output sequence. It will get $P_0 + P_1 = 1$ in the output sequence. Based on (11), the formulas for calculating the new probability of the output sequence are derived

$$P_1^{new} = \frac{P_A^1 P_B^1 P_{ch}^1}{P_A^1 P_B^1 P_{ch}^1 + (1-P_A^1)(1-P_B^1)(1-P_{ch}^1)} = \frac{P_A^1 P_B^1 P_{ch}^1}{P_A^1 P_B^1 P_{ch}^1 + P_A^0 P_B^0 P_{ch}^0}, \tag{13}$$

$$P_0^{new} = \frac{P_A^0 P_B^0 P_{ch}^0}{P_A^0 P_B^0 P_{ch}^0 + (1-P_A^0)(1-P_B^0)(1-P_{ch}^0)} = \frac{P_A^0 P_B^0 P_{ch}^0}{P_A^0 P_B^0 P_{ch}^0 + P_A^1 P_B^1 P_{ch}^1}. \tag{14}$$

3.3. Detailed Update Process with Regeneration

Based on the above investigation on stochastic VNs, the update steps of VNs in our decoder are as follows:

Step 1: The first step is similar to the VN update part in traditional hard-decision decoding. Regardless of the sequence of the target edge that needs to be updated, a majority decision is made on the sequence of the remaining edges and the initial input sequence to obtain the initial output sequence. Consider an example with $L = 6$ and the VN degree 3. Then when we update the sequence of the third edge e_3 of current VN. The process of majority decision is shown in Table 2.

Table 2. Majority decision of the sequences

Input			Output
e_1	e_2	S_{ch}	e_3
0	0	0	0
1	1	1	1

0	1	0	0
0	0	0	0
0	0	0	0
0	1	1	1
0	0	1	0
1	1	0	1

In Table 2, the probability of 0 judged by 000 in the output is $P_0^{old} = \frac{3}{8}$, and the probability of 1 judged by 111 in the output is $P_1^{old} = \frac{1}{8}$. These old probabilities cannot add up to 1.

Step 2: Continuing to take Table 2 as an example, calculate the number of zeros judged by 000 in the sequence of e_3, that is, the number of zeros which are regular state in the output sequence. This number is recorded as $Num_0 = 3$. Similarly, the number of ones judged by 111, that is, the number of ones which are regular state in the output sequence. This number is recorded as $Num_1 = 1$. According to (13) and (14), new P_0 and P_1 of the sequence of e_3 can be obtained, as shown in (15), and (16).

$$P_0^{new} = \frac{Num_0}{Num_0 + Num_1} = \frac{3}{4} \tag{15}$$

$$P_1^{new} = \frac{Num_1}{Num_0 + Num_1} = \frac{1}{4} \tag{16}$$

Step 3: According to P_0^{new} and P_1^{new}, the number of 0 and 1 in the original output sequence will be changed. But we only change 0 and 1 which are not judged by 000/111. When P_0^{new} is greater than the original P_0^{old}, a part of 1 which are not judged by 111 in the output sequence will be changed to 0. Conversely, when P_1^{new} is greater than the original P_1^{old}, a part of 0 which are not judged by 000 in the output sequence will be changed to 1. An example of regeneration of the output in Table 2 can be seen in Table 3.

In this example, with $L = 8$, the regenerated number of 0 is 6 according to the probability in (15) and regenerated number of 1 is 2 according to the probability in (16). Comparing to the output sequence in Table 2, it is found that the number of 1 should be decreased from 3 to 2. There are two 1 which

are not judged by 111 in the example, and one of them should be changed. Here, the 1 judged by 011 is changed to 0 (the bolded value).

Table 3. An example of regeneration

Input			Output	Regenerated Output
e_1	e_2	S_{ch}	e_3	e_3^{new}
0	0	0	0	0
1	1	1	1	1
0	1	0	0	0
0	0	0	0	0
0	0	0	0	0
0	1	1	1	**0**
0	0	1	0	0
1	1	0	1	1

Step 4: When generating the final result, it is similar to updating the edge of VNs. Using a new example in Table 4, we will bring all the edges and initial sequence into the majority decision in the step of generating the final result. Then, comparing the number of results judged by 0000 and the number of the result judged by 1111, the example of final decision process is shown in Table. 4.

Table 4. Decision part of final result

e_1	e_2	e_3	S_{ch}	Output
0	0	0	0	**0**
1	1	1	1	**1**
0	1	0	0	0
0	0	0	0	**0**

0	0	0	0	**0**
0	1	1	1	1
0	0	1	0	0
1	1	0	1	1

In Table 4, the bolded values in the output sequence are 0 and 1 which are in regular state. Since the number of bolded zeros is more than the bolded ones, the final output of this example VN is 0.

3.4. Simulation Performance

3.4.1. Performance vs. Stochastic Sequence Length *L*

For our stochastic decoding method, the most important factor affecting its performance is the length of the stochastic sequence. Because in the initial stage of decoding, we need to generate the stochastic sequences based on the output probability from the channel. When the length of the sequence is longer, the distribution of 0 and 1 is more in line with the output probability of the channel, that is to say, the reliability of such a sequence is higher. If a highly reliable sequence is used as input in iterative decoding, the probability of the output sequence will be closer to the output value of BP decoding, so as to achieve better decoding performance.

In this simulation, the simulation time of each SNR is too long to observe the situation of all SNR. Thus, we select a fixed SNR, let SNR=2.5dB. And select the sequence length *L* distribution range from 100 to 1000 with an interval of 100. The simulation result is shown in Fig. 3.4.

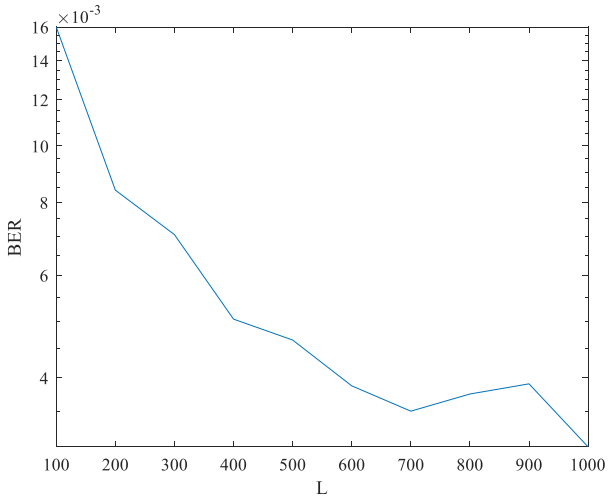


Fig. 3.4 BER vs. sequence length L, SNR=2.5dB

From Fig. 3.4, it is not difficult to see that, as the length of the sequence increases, the BER is continuously reduced. When the sequence length L changes from 100 to 400, the BER will decrease rapidly. When the sequence length L changes from 400 to 1000, the downward trend of the BER gradually tends to be flat.

Then, as shown in Fig. 3.5, we select the SNR from 1dB to 3dB to observe the changes in the bit error rate curves of different sequence length L.

In Fig. 3.5, when the sequence length L is greater than 400, the performance of the stochastic decoder is less affected by the sequence length L. But in general, the longer stochastic sequence will still improve the performance of the decoder.

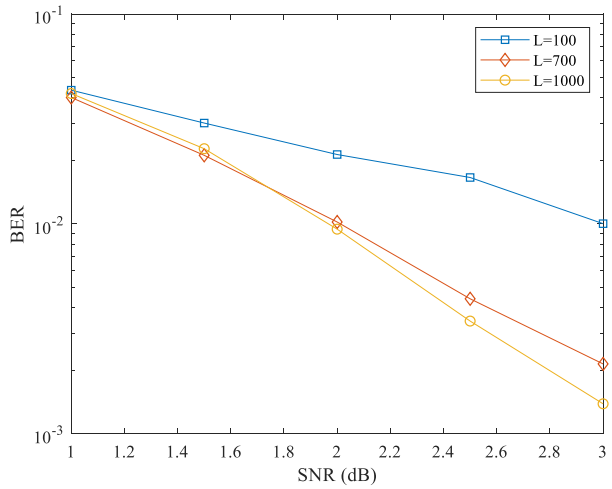


Fig. 3.5 BER of different sequence length L=100, L=700 and L=1000

3.4.2. Comparison of Stochastic Decoding and BP Decoding

In our simulation, we firstly test our decoder based on the (1024, 3, 6) LDPC code, and observe the difference between binary stochastic decoder and BP decoder. In order to achieve better decoding results, the sequence length of stochastic decoding is $L = 1000$. The simulation result is shown in Fig. 3.6.

In Fig. 3.6 and Fig. 3.7, we can clearly see that the regeneration method can greatly improve the performance of binary stochastic decoding.

From Fig. 3.6, we can also clearly see that for long LDPC codes, the performance of BP decoding and the performance of stochastic decoding are similar at low SNR. At high SNR, the performance of BP decoding will be much better than that of stochastic decoding. This also proves that BP decoding is suitable for decoding long codes, and as the code length is longer, the performance will approach the Shannon limit.

From Fig. 3.7, we can clearly see that the decoding performance of BP decoding indeed decreases for short LDPC codes. In the low SNR region, the performance of stochastic decoding is slightly better than that of BP decoding, while in the high SNR region, the performance of stochastic decoding is slightly worse than BP decoding. Hence this method still needs to be improved.

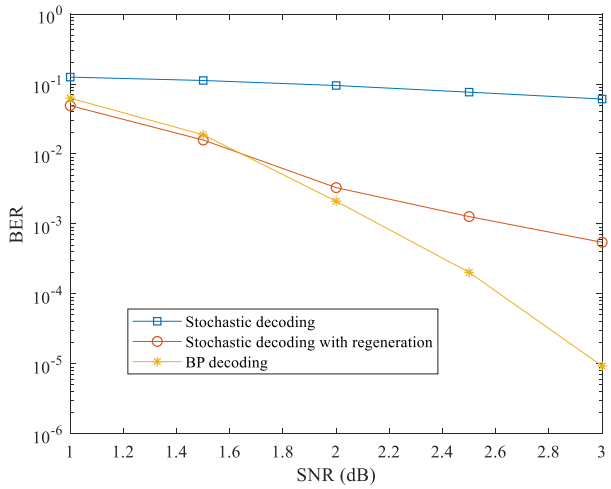


Fig. 3.6 Comparison of different decoding methods, $L = 1000$, (1024, 3, 6) LDPC code

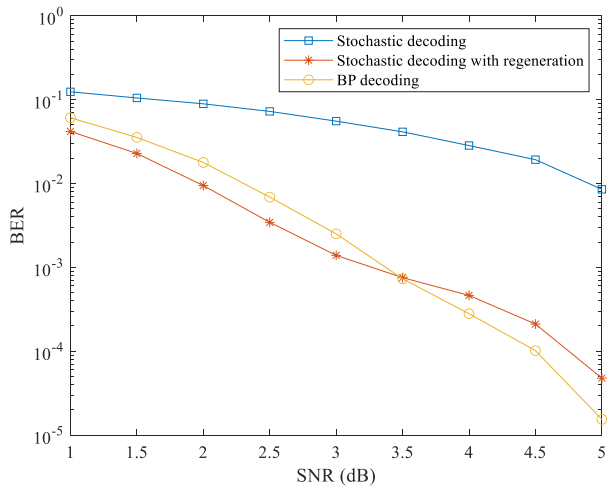


Fig. 3.7 Comparison of different decoding methods, $L = 1000$, (126, 3, 6) LDPC code

4. Stochastic List Decoding

Elias proposed list decoding in the 1950s in coding theory [18]. The main idea behind list decoding is that the decoding algorithm will not only output one decoding result but output multiple decoding results to form a list. Compared with the upper limit of error correction for unique decoding, list decoding can correct more errors. Although list decoding takes the similar decision procedure to ML decoding, there is still a little difference between them. In the typical ML decoding, the decoder will output a decoding result that is closest to the received vector among all of the possible codewords as the final result based on the Euclidean distance. However, list decoding only correlates the received vector with a part of possible codewords instead of correlating with all of the possible codewords to do the final decision. Thus, list decoding has a lower decoding complexity than ML decoding while it still can take the advantage of ML decision.

Inspired by [6], our stochastic decoding can be combined with list decoding to take the advantage of the ML decision part. In this section, the performance of stochastic list decoding with hard input (0/1 bit) and stochastic list decoding with soft input (designed LLR) will be discussed, respectively. Hard stochastic list decoding is proposed on the basis of the stochastic decoding with regeneration method which is mentioned in Section 3. Soft stochastic list decoding is proposed based on the advantage of the BP decoding algorithm.

4.1. Stochastic List Decoding with Hard Input

As mentioned in Section 3, the stochastic sequences can be generated to form an input table in the first step of stochastic list decoding. Then, each row of the input table is taken into the designed decoder (Tanner graph). After iterative decoding, the decoder will output a result table before the final decision. In the decision part of the current decoding method, the hard ML decision method is used due to the hard input. The hard ML decision means that the result table after decoding just includes 0 and 1, and the row of this binary result table is correlated with the received symbol after the BPSK modulation. The general ML decision process can be expressed as

$$\hat{\mathbf{v}} = \arg \max_{i \in 1,\ldots,L} \left(\hat{\mathbf{v}}_i \cdot \mathbf{r}^{\mathrm{T}} \right). \tag{17}$$

Here, $\hat{\mathbf{v}}_1$ to $\hat{\mathbf{v}}_L$ are the possible output codewords of our stochastic decoder.

The performance of stochastic list decoding will be compared with stochastic decoding with regeneration method and BP decoding. The sequence length of the stochastic sequence is $L = 1000$. The simulation result is shown in Fig. 4.1.

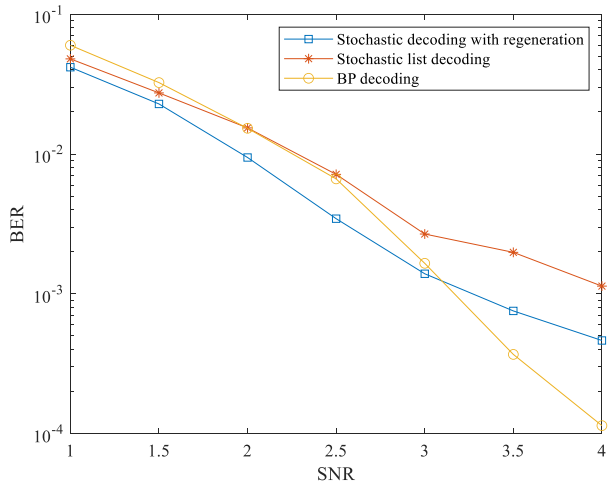


Fig. 4.1 Performance of stochastic list decoding, stochastic decoding and BP decoding, $L = 1000$

One might think that the performance of stochastic list decoding should be better than stochastic decoding with regeneration, because the list method replaces one result with multiple candidate results, which makes the probability of successful decoding higher. However, from Fig. 4.1, we can find that the performance of stochastic list decoding is even worse than stochastic decoding with regeneration. In the high SNR region, the gap between BP decoding and stochastic list decoding becomes large. It's not as good as the expected performance. This bad performance is mainly because we only let one row of the result table to join in the correlation. In every correlation, only one codeword joining cannot provide the high reliability, so it is too weak to decode the code successfully for stochastic list decoding.

4.2. Enhanced Stochastic List Decoding with Hard Input

Based on the conclusion in Section 4.1, the reliability of each decoded codeword needs to be enhanced. Thus, the output after iterative decoding needs to be pre-processed before the decision part. In order to make these

outputs more robust, every five bits of each original output sequences are combined to generate an enhanced bit with majority decision. Fig. 4.2 shows an example of the process of enhancement. In Fig. 4.2, the length of sequence is assumed to be 15 and V1-V6 represent the six VNs. After the process of the enhancement, a new output table can be obtained. Every bit in this table is generated from the combination of the five bits in the original output table.

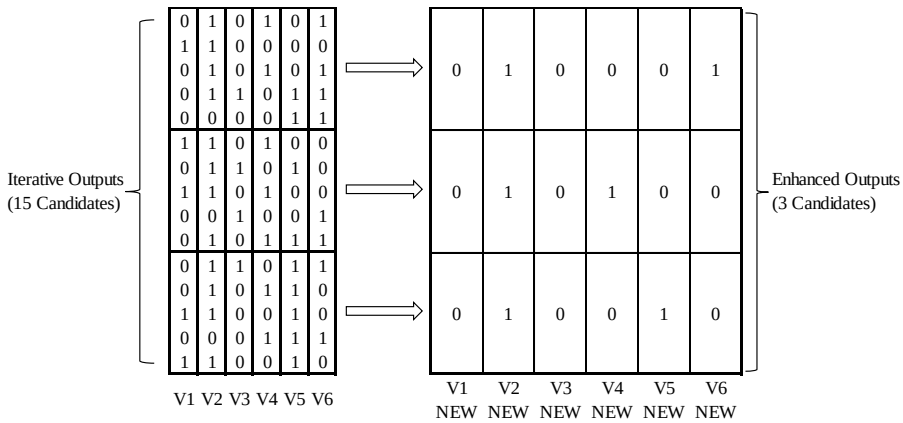


Fig. 4.2 An example of the process of the enhancement

In the simulation of this decoding method, the sequence length L is the same as the simulation in Section 4.1. Fig. 4.3 shows the comparison of the enhanced stochastic list decoding, stochastic list decoding and BP decoding.

Unlike the expectation, the performance of the enhanced stochastic list decoding is similar to original stochastic list decoding in the low SNR region, and just a little better than the original one in the high SNR region. This bad performance may be caused by the iterative hard-decision decoding. Typically, hard-decision decoding always has a worse performance than the BP decoding in the iterative decoding part [19].

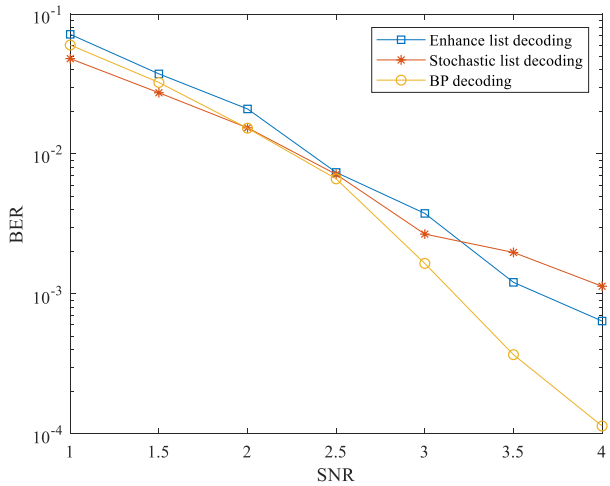


Fig. 4.3 Performance of enhanced stochastic list decoding, stochastic list decoding and BP decoding, $L = 1000$

4.3. Stochastic List Decoding with Soft Input

According to the bad performance of two decoding methods in Section 4.1 and Section 4.2, a stochastic list decoding with soft input becomes the new direction to study. In order to combine the BP decoding algorithm instead of the hard-decision decoding with the list method, we need to adjust the input stochastic sequence to let the message passing between the CNs and VNs be soft values (LLR) rather than binary symbols. When the input table is generated, we can convert every binary symbol to their corresponding LLR. Equations (18) and (19) are derived from (2) to realize the transformation of the 0 and 1 respectively.

$$0 \to L_0 = \ln\frac{p(x=0|s=0)}{p(x=1|s=0)} = \ln\frac{1}{0} = \infty \tag{18}$$

$$1 \to L_1 = \ln\frac{p(x=0|s=1)}{p(x=1|s=1)} = \ln\frac{0}{1} = -\infty \tag{19}$$

Here, s is the existing value of the current bit which needs to be transformed and x is the possible value that the current bit could be. $p(x=0|s=0)$ stands for the posterior probability of current bit equal to 0 under the existing condition of current bit equal to 0. Here, only one bit needs

to be transformed each time. Thus, if the current bit is 0, then $p(x = 0|s = 0)$ is 1. In the same way, if the current bit is 1, then the $p(x = 1|s = 1)$ is 1. The positive infinity L_{max} in (18) and the negative infinity $-L_{max}$ in (19) are represented with the empirical value $L_{max} = 8$ and $-L_{max} = -8$, the choice of which will be further discussed in Section 5. Fig. 4.4 shows an example of the transformation of the binary input table to the log-likelihood ratio input table

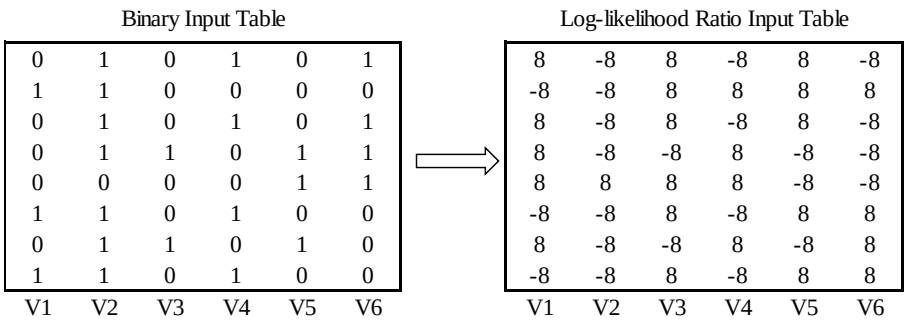


Fig. 4.4 An example of the transformation from binary input table to log-likelihood ratio input table

Then, each row of the log-likelihood ratio input table is taken into the iterative decoding part with the BP decoding algorithm. After simulation with the same sequence length of $L = 1000$ as the ones used in Section 4.2 and Section 4.1, the comparison of the performance of stochastic list decoding with soft input, enhanced stochastic list decoding with hard input, original stochastic list decoding, stochastic decoding with regeneration and BP decoding can be seen in Fig. 4.5. We can clearly find that the stochastic list decoding with soft input can significantly improve the performance of the stochastic decoder.

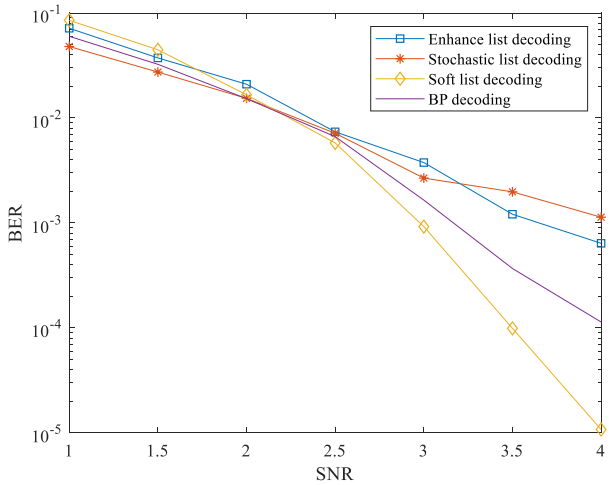


Fig. 4.5 Comparison of different stochastic list decoding and BP decoding, $L = 1000$. L_{max}=8 in soft list decoding

5. Stochastic Decoding with Non-Binary Quantized Input Sequence

In Section 3, a stochastic decoding with binary input stochastic sequence is introduced, while this decoding method doesn't achieve a very good performance with a sequence length of $L = 1000$. Even though the ML decision is applied to the output list of the binary stochastic decoding in Section 4.1, the performance of this decoding method still has a severe loss. However, the performance of the stochastic decoder gets a little improvement with enhancement method in the high SNR region in Section 4.2, and the performance gets much better than BP decoding with the transformation of the binary input sequences in Section 4.3. Thus, inspired by Section 4.2 and Section 4.3, the stochastic decoding with non-binary quantized input sequence is proposed in this section.

5.1. Detailed Decoding Process

The decoding process of stochastic decoding with non-binary quantized input sequence can be divided into three steps. The first step is the generation of the non-binary quantized sequence on the basis of the binary stochastic sequence which is mentioned in Section 2.3.3 and Section 3.1. The second step is to transform the non-binary symbol to the LLR and do the iterative decoding with the BP decoding algorithm. The third step is the decision process which will be operated with soft ML decision, hard ML decision and average decision separately. The decoding process can be seen in Fig. 5.1.

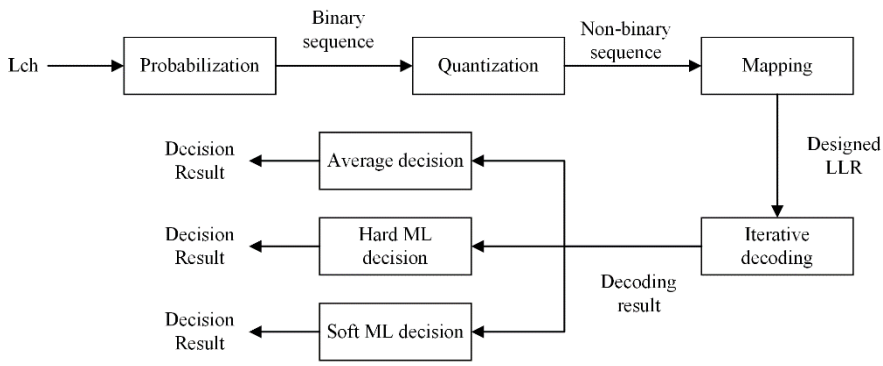


Fig. 5.1 The decoding process of the non-binary stochastic decoding

5.1.1. Generation of the Non-Binary Quantized Input Sequence

At the beginning of the generation of the non-binary quantized input sequence, the binary quantized input sequence is divided into several sub-sequences on the basis of the enhancement method mentioned in Section 4.2. These sub-sequences will be transferred to the non-binary symbol with the quantization method. Here, the number of the bits combined each time (length of the sub-sequence) is called as bit width which is defined as w. The length of the original binary stochastic sequence is defined as L_b and the length of the resulted non-binary stochastic sequence is defined as L_s. The relationship of these three parameters is

$$L_b = w \cdot L_s \ . \tag{20}$$

Then, in order to get the non-binary symbol S_i, the quantization method is operated by adding all of the bits up in one sub-sequence rather than using the majority decision on these bits. Table 5 shows an example of the quantization of the bit width of $w = 3$.

Table 5. An example of the quantization of the 3-bit sub-sequence

Possible Sub-sequences	0	0	0	1	1	1	0	1
	0	0	1	0	1	0	1	1
	0	1	0	0	0	1	1	1
$\boldsymbol{S_i}$	0	1			2			3
$\boldsymbol{P_1}$	0	$\frac{1}{3}$			$\frac{2}{3}$			1

In Table 5, 3-bit means that each sub-sequence is composed of 3 bits, which leads to 8 possible sub-sequences with $w = 3$. Besides, P_1 represents the probability of 1 in the 3-bit sub-sequence. In other words, bit width w will cause 2^w possible sub-sequences and the value of the quantized symbol will be an integer from 0 to w. Then, the non-binary sequence can be generated according to the quantization method. Fig. 5.2 shows an example of the generation of a non-binary stochastic sequence with $w = 3$.

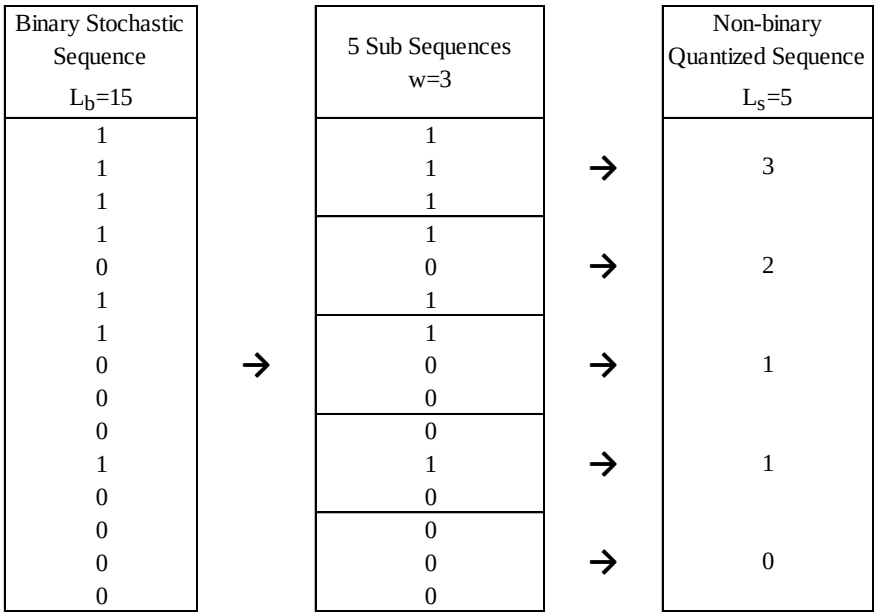


Fig. 5.2 An example of the generation of the 3-bit quantized sequence

In Fig. 5.2, the probability of 1 in the binary stochastic sequence is $P_b^1 = \frac{7}{15}$. In the non-binary quantized sequence, the probability of the 1 in the possible sub-sequence is also shown in the Table 5.

Based on Table 5, the probability of 1 in the non-binary quantized sequence is $P_{nb}^1 = \frac{1}{5} \cdot 0 + \frac{2}{5} \cdot \frac{1}{3} + \frac{1}{5} \cdot \frac{2}{3} + \frac{1}{5} \cdot 1 = \frac{7}{15}$. Hence, probabilities of 1 in binary stochastic sequence and non-binary stochastic sequence are the same.

5.1.2. Transformation of the Non-Binary Quantized Input Sequence

Inspired by stochastic list decoding with soft input in Section 4.3, the quantized symbols in the non-binary quantized sequence need to be transformed to the LLR in order to use the BP algorithm in the iterative decoding part.

Based on (18) and (19), the LLR of the corresponding quantized symbols in the non-binary input sequence can be calculated with

$$L_{S_i} = \ln \frac{P(x = 0|S_i)}{P(x = 1|S_i)}. \tag{21}$$

A little unlike (18) and (19), the S_i in (21) is composed of multiple binary bits which belong to the original binary stochastic sequence rather than equaling to the bit itself, and x is a bit included in the S_i. Thus, according to the Bayes' theorem, the posterior probability of the binary bit x under the condition of the corresponding symbol S_i can be described with

$$P(x=0|S_i) = \frac{P(S_i|x=0)P(x=0)}{P(S_i)}, \tag{22}$$

$$P(x=1|S_i) = \frac{P(S_i|x=1)P(x=1)}{P(S_i)}. \tag{23}$$

In (22) and (23), $P(x=0)$ equals to P_0 which is the priori probability of 0 bit in the original binary sequence and $P(x=1)$ equals to P_1. $P(S_i|\mathrm{x}=0)$ is the likelihood of $x=0$ based on observing S_i under the condition of one bit included in S_i equaling to 0. This likelihood can be calculated with

$$P(S_i|x=0) = \begin{cases} 0 & , \quad S_i = w \\ \binom{w-1}{S_i} P_0^{w-S_i-1} P_1^{S_i}, & S_i \neq w \end{cases}. \tag{24}$$

In the same way, $P(S_i|\mathrm{x}=1)$ can be calculated with

$$P(S_i|x=1) = \begin{cases} 0 & , \quad S_i = 0 \\ \binom{w-1}{S_i-1} P_0^{w-S_i} P_1^{S_i-1}, & S_i \neq 0 \end{cases}. \tag{25}$$

$P(S_i)$ in (22) and (23) is the probability of the symbols which can be calculated with

$$P(S_i) = \binom{w}{S_i} P_0^{w-S_i} P_1^{S_i}. \tag{26}$$

Based on (24) – (26), (22) and (23) can be rewritten to

$$P(x=0|S_i) = \frac{w-S_i}{w}, \tag{27}$$

$$P(x=1|S_i) = \frac{S_i}{w}. \tag{28}$$

Thus, the LLR of the corresponding quantized symbols of (21) can be rewritten to

$$L_{S_i} = \ln \frac{P(x = 0|S_i)}{P(x = 1|S_i)} = \ln \frac{w - s_i}{s_i}. \quad (29)$$

What should be noted in (29) is that the value of bit width w cannot be an even value, since when the quantized symbol S_i is equal to $\frac{w}{2}$, the corresponding LLR will be 0. If the LLR is 0, it will cause the problem in the updated process of CN. As (4) shown in Section 2.3.1, value of 0 will force the result of the multiplication be 0 and force all of the updated result to be 0 with the iteration going.

Then, taking the non-binary quantized sequence with a bit width of $w = 5$ as an example (w=5 is an odd value), the corresponding L_{S_i} of the quantized symbols can be seen in Table 6.

Table 6. An example of transformantion to L_{S_i}

Non-binary Quantized Sequence	Corresponding L_{S_i} Sequence
0	$+\infty$
1	$\ln \frac{1}{4}$
2	$\ln \frac{2}{3}$
3	$-\ln \frac{2}{3}$
4	$-\ln \frac{1}{4}$
5	$-\infty$

In Table 6, S_i =5 in the non-binary quantized sequence means the corresponding sub-sequence of this symbol is full-one and S_i=0 means the corresponding sequence is full-zero. These two kinds of sub-sequences will cause the corresponding LLR be positive infinity and negative infinity respectively. In order to realize the BP algorithm in the iterative decoding part, these infinity values should be replaced with L_{max} and $-L_{max}$. In Section 4.3, an empirical value of 8 is used to represent the infinity. However, it seems that different values will cause the different performance of the

stochastic decoder. Thus, the detailed choice and analysis of this value will be discussed later.

5.1.3. Different Decision Methods

After iterative decoding, the output table should be taken into the decision part, which is designed to decode the sequence. In this section, the averaged decision method, hard-decision method, and soft-decision method are introduced.

The averaged decision is the simplest decision method which is similar to the majority decision method. In the results table after iterative decoding, each final output of VN is represented by a stochastic output sequence composed of non-binary LLR values. Hence, the majority decision method cannot simply be used, instead, the averaged method can be used. In this method, summing up the whole sequence, when the sum value is greater than 0, the judgment result is 0 and on the contrary, the judgment result is 1.

As mentioned in Section 4, the ML decision method has a good performance typically. Two kinds of ML decision methods can be taken into consideration which are the hard-decision method and the soft-decision method. In the hard-decision method, firstly to transform the result table after iterative decoding when L_{out} of the iterative output is greater than 0, the result should be replaced with 0, and on the contrary, the result should be replaced with 1. Composed of the binary sequence, the transformed new result table can be used to calculate the optimal result with ML hard-decision as mentioned in Section 4.1. In the soft-decision method, the correlation between the LLR sequence after iterative decoding and the original LLR sequence of the channel output is calculated to make an optimal decision.

Based on the simulation results and the empirical results, it can be supposed that the soft-decision method will have a better performance. So, the soft-decision method is firstly used to study the performance of stochastic decoder with the non-binary input sequence, while the performance of other decision methods will be discussed in Section 5.2.4.

5.2. Simulation Performance

Based on the decoding algorithm, three parameters have the impact on the performance of the whole decoder, which are quantized value L_{max}, bit width w, and sequence length L_s. Here, the quantized value is the value which can stand for the positive infinity and negative infinity of LLR as mentioned in Section 5.1.2. Then, bit width w is a variable that should be taken into consideration in the simulation, since it can change the diversity of the

sequence. Besides, non-binary sequence length L_s also has an impact on the complexity and the gain of the decoder. Consequently, after determining the suitable parameters, the simulation can be designed to study the performance of different decoding methods. And all of the simulation in this section will be operated with the (126,3,6) LDPC code.

5.2.1. Influence of Quantized Value L_{max}

Firstly, how the quantized value L_{max} influences the performance of the decoder should be studied. In this simulation, the initial bit width is $w = 7$ and the sequence length is $L_s = 1000$. Different quantized values L_{max} have been chosen randomly in the interval of [4,25] to generate a bit error rate curve with SNR of 1 to 4 and the simulation result is shown in Fig. 5.3.

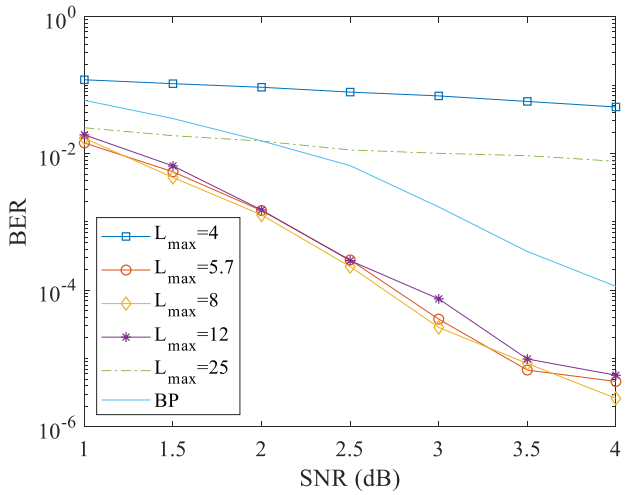


Fig. 5.3 Comparison with different quantized values L_{max}, $w = 7$, $L_s = 1000$, BER

In Fig. 5.3, the decoders with $L_{max} = 4$ and $L_{max} = 25$ show a bad performance compared to the BP decoding algorithm while other decoders show a similar performance which is much better than the BP decoding algorithm, and the $L_{max} = 8$ is the best among them. Thus, $L_{max} = 8$ can be used in the continued simulation to maintain a good performance.

5.2.2. Influence of Bit Width w

In this section, the impact of the bit width w is studied with $L_{max} = 8$ as mentioned above, and the sequence length of $L_s = 1000$. The simulation result of this experiment is shown in Fig. 5.4.

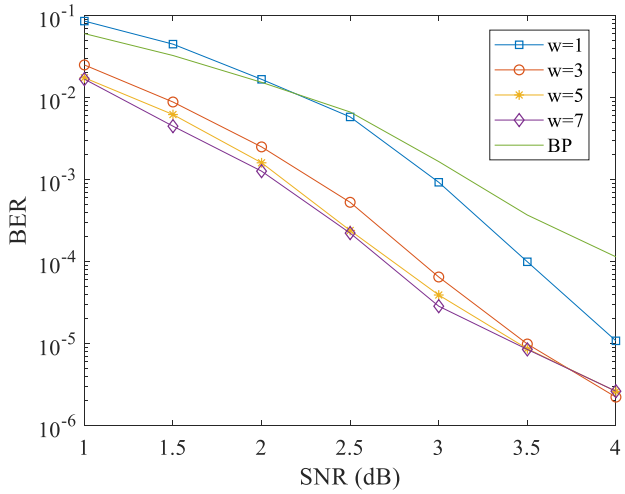


Fig. 5.4 Comparison with different bit widths w, $L_{max} = 8$, $L_s = 1000$, BER

In Fig. 5.4, the performance of the decoder with the non-binary input is much better than the one using binary input. Besides, although the performance becomes better with increasing bit width w. Since the step of quantifying the sequence is completed before the iteration, a large bit width w can be used in non-binary sequence generation without the high cost. So, 7-bit is chosen to get better performance.

5.2.3. Influence of Non-Binary Sequence Length L_s

Based on the simulated results in Section 5.2.1 and Section 5.2.2, the quantized value can be $L_{max} = 8$ and the bit width can be $w = 7$ to study how the non-binary sequence length L_s affects the bit error rate. The simulation result is shown in Fig. 5.5.

In Fig. 5.5, with the sequence length L_s increased, the bit error rate performance improves. However, when a larger sequence length L_s is used, the decoder pays the price of increased complexity. As can be shown in the Fig. 5.5, even though the complexity increases 10 times, the performance of

the decoder with $L_s = 1000$ just improves a little compared to the decoder with L_s =100. So, considering the balance between complexity and performance, a sequence length of L_s=100 can be chosen to reduce the computational resource.

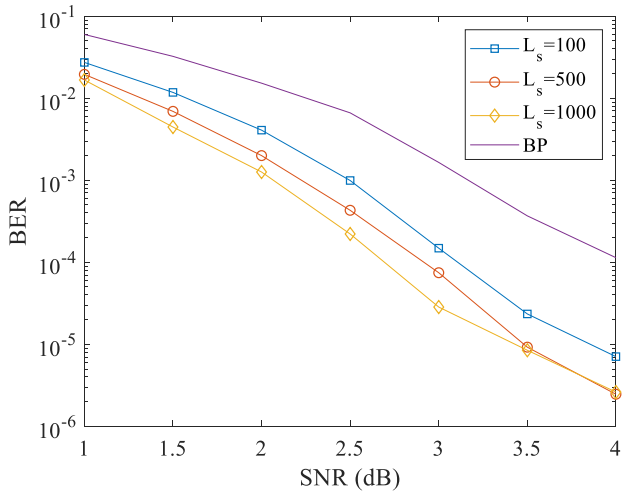


Fig. 5.5 Comparison with different quantized input sequence lengths L_s, $L_{max} = 8$, $w = 7$, BER

5.2.4. Comparison of Different Decision Methods

Based on the simulation result above, with $L_s = 100$, $w = 7$, and $L_{max} = 8$, the performance of the stochastic decoder is better than the BP decoder. Then we still need to compare the performance of the others decision methods to decide which decision method is the best one. By using the same parameters as the soft-decision method, the performance of the hard-decision method and the average decision method can be simulated. The simulation results are shown in Fig. 5.6.

Based on the simulation results, the decoder using the average decision method has the worst performance which is worse than BP decoding, and the decoder with the hard-decision method has a higher bit error rate than the decoder with the soft-decision. Hence, the soft-decision method can be chosen as the final decision method for the decoder to get the best performance.

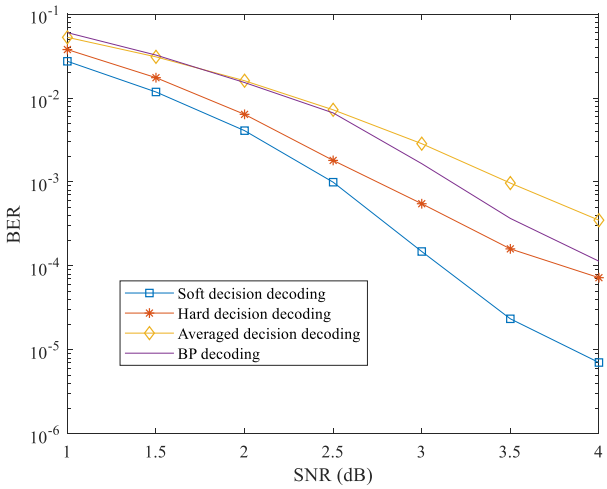


Fig. 5.6 Comparison of different decision methods, $L_s = 100$, $w = 7$, and $L_{max} = 8$, BER

5.3. Complexity Reduction

In Section 5.2, how the quantized value L_{max}, bit width w, input non-binary sequence length L_s, and the decision method influence the performance of the decoding has been studied. The decoder that can achieve the best performance with low complexity is the decoder with the soft ML decision method, $L_{max} = 8$, $w = 7$, and $L_s = 100$. However, typically when a quantized sequence length of $L_s = 100$ is used in a stochastic decoding method, 100 times the price of the complexity is paid compared to the BP decoding algorithm. So, how to reduce the complexity should be considered so that the computational resources of the decoder can be reduced.

5.3.1. Complexity Reduction with Bit Width of $w = 7$

Based on the simulation result in Section 5.2.3, the length of the non-binary input sequence can be reduced to decrease the complexity. Aiming to balance the complexity and the performance of the decoding error rate, four different parameters which are bit error rate, frame error rate, the average number of iterations, and the computational complexity, can be observed to evaluate the performance and the complexity of the decoder. These four parameters will be introduced below.

Showing how the algorithm performs in decoding, the bit error rate is the probability of bit transmission errors during data transmission, while the frame error rate refers to the probability of frame transmission errors. Besides, the average number of iterations which is defined as I^{av} calculates the average iteration times in the simulation, and computational complexity, which can intuitively observe complexity, counts the number of operations in the iteration decoding part and the decision part. With average iteration times I^{av} and computational complexity, the complexity of decoding can be evaluated. Among all these parameters, the BER, FER, and average iteration times I^{av} can be directly observed in the simulation result, but the computational complexity should be calculated by analyzing the algorithm.

To analyze the computational complexity of the decoding algorithm in this paper, the computational complexity of the BP algorithm for one frame decoding can be calculated as a standard, based on the BP decoding algorithm introduced in Section 2.3.1. The operands for one frame of BP algorithm is shown in Table 7.

Table 7. The computational complexity of BP decoding algorithm

Operation Calculation of SPA decoding in One Iteraion

Operation	Check node	Variable Node
Multiplication	60	
Add		9
Hyperbolic Tangent	30	
Inverse Hyperbolic	6	
Total in One Iteration	6048	1134

Total Operation of all Iterations: $7182 \cdot I^{av}$

When the BP algorithm is used to decode a frame with a length of 126 bits, the sum of all the operations that need to be performed is

$$Operand(BP) = 7182 \cdot I^{av}. \quad (28)$$

With the operands of the algorithm, the computational complexity can be observed.

For the reason that the stochastic algorithm introduced in Section 5 uses the same iteration method with BP decoding, the operand of the stochastic algorithm can be calculated based on the operand of the BP decoding algorithm. Table 8 shows the statistics of the stochastic algorithm operand, and L_s refers to the length of the non-binary input sequence.

Table 8. The computational complexity of stochastic decoding algorithm

Operation calculation of the generation of the non-binary quantized sequence		
Operation	Operand	
Multiplication	1	
Exponentiation	2	
Add	1	
Mapping	L_S	
Total	$126\cdot(4+L_S)$	
Operation Calculation of BP Decoding in One Iteraion		
Operation	Check node	Variable Node
Total in One Iteration	$6048\cdot L_S$	$1134\cdot L_S$
Operation Calculation of ML Decision		
Operation	Times	
Multiplication	$126\cdot L_S$	
Add	$125\cdot L_S$	
Comparison	1	
Total in One Iteration	$251\cdot L_S+1$	
Total Operation: $(7433\cdot L_S+1)\cdot I^{av}+126\cdot(4+L_S)$		

The operand of the stochastic decoding is

$$Operand(STO) = (7433 \cdot L_s + 1) \cdot I^{av} + 126 \cdot (4 + L_s) . \quad (30)$$

With these four parameters, how to balance the performance and the complexity of the algorithm can be studied.

The simulations are designed to see how the system performs with a sequence length of $L_s = 10$, $L_s = 20$, and $L_s = 100$, in which $w = 7$ and $L_{max} = 8$ are used. Fig. 5.7-5.10 show BER, FER, the average number of iterations I^{av}, and computational complexity vs. SNR curve in the simulation. From Fig. 5.7 and Fig. 5.8, with an input sequence length of $L_s = 10$, the stochastic decoding with non-binary input sequences shows similar performance to the BP decoding in BER, while the decoding performance even worse than the BP decoding when considering the FER. Consequently, to obtain good performance, the lowest sequence length that can be used is $L_s = 20$ with the bit width of $w = 7$ and a quantized value of $L_{max} = 8$.

In Fig. 5.7 and Fig. 5.8, the stochastic decoder with an input sequence length of $L_s = 20$ gets an approximately 9 times enhancement compared to the BP algorithm at SNR=4dB, while in Fig. 5.9 and Fig. 5.10, compared to the BP algorithm, even though it has a smaller average iteration time I^{av}, the stochastic decoder has more than ten times the computational complexity. Hence, the decoder needs to be improved to reduce the complexity and increase the performance.

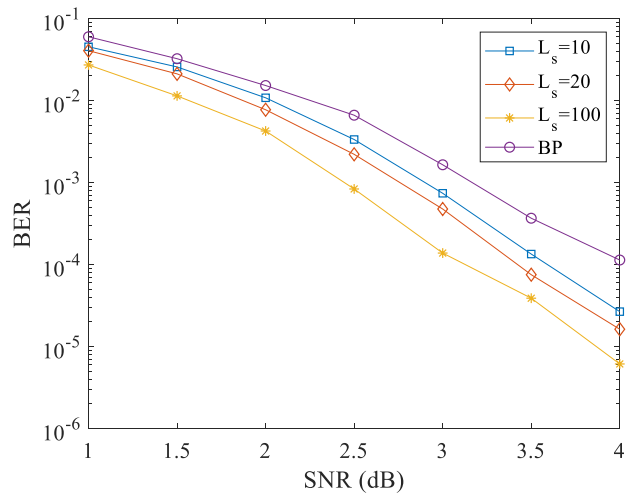


Fig. 5.7 Comparison of different non-binary input sequence lengths L_s, $w = 7$, $L_{max} = 8$, BER

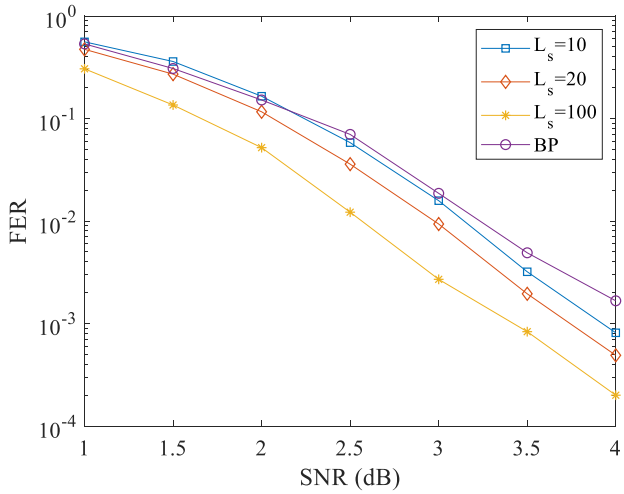


Fig. 5.8 Comparison of different non-binary input sequence lengths L_s, $w = 7$, $L_{max} = 8$, FER

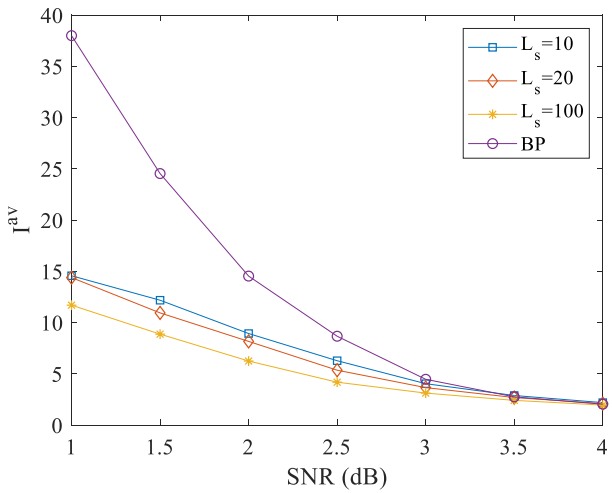


Fig. 5.9 Comparison of different non-binary input sequence lengths L_s, $w = 7$, $L_{max} = 8$, average iteration times I^{av}

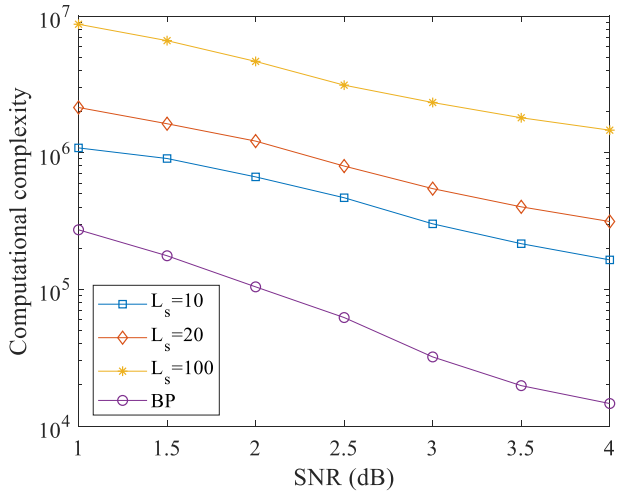


Fig. 5.10 Comparison of different non-binary input sequence lengths L_s, $w = 7$, $L_{max} = 8$, computational complexity

5.3.2. Further Complexity Reduction

To further reduce the complexity, how to improve the performance of the algorithm without increasing the complexity too much should be considered. Based on the result in Section 5.2.2, the performance of the algorithm improves with the increasing bit width w. As can be seen in Table 8, the computational complexity of the non-binary sequence generation part in the algorithm is relatively low compared to other parts. Based on (29) the computational complexity of the algorithm is not affected much by the bit width w of the input sequence. Hence, the bit width w can be increased to improve the performance of the algorithm without increasing the computational complexity. To study whether the performance can further improve, 7-bit, 15-bit, and 31-bit can be chosen as the bit width w in the simulation with a non-binary input sequence length of $L_s = 20$ and the quantized value of $L_{max} = 8$. The simulation results are shown in Fig. 5.11-Fig. 5.14.

Fig. 5.11 and Fig. 5.12 show the comparison of the BP decoding method and the stochastic algorithm in BER and FER. With the same sequence length of $L_s = 20$, 7-bit, 15-bit, and 31-bit all have a better performance than BP decoding, among which the decoder with a bit width of $w = 15$ shows the best performance.

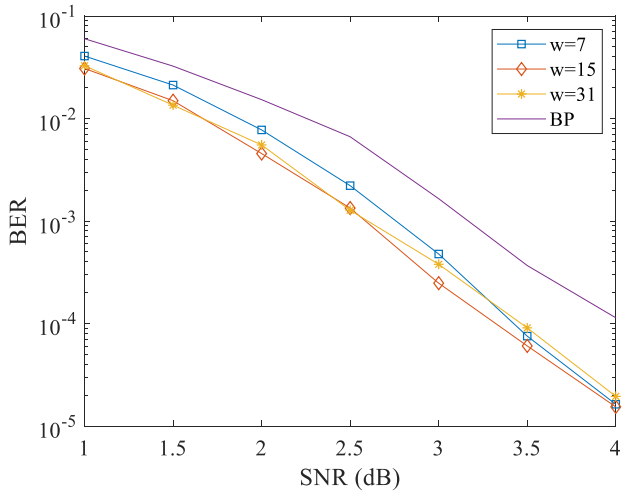


Fig. 5.11 Comparison of different bit widths w, $L_s = 20$, $L_{max} = 8$, BER

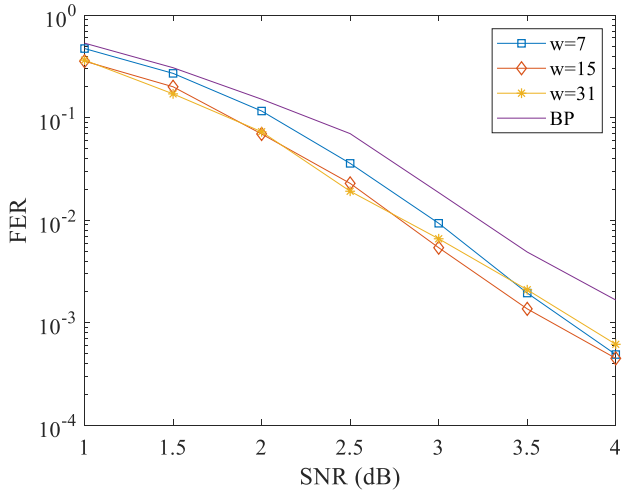


Fig. 5.12 Comparison of different bit widths w, $L_s = 20$, $L_{max} = 8$, FER

When considering the computational complexity in Fig. 5.13 and Fig. 5.14, with the bit width w increased, the average iteration time I^{av} decreases, leading to a slight decrease in the complexity. However, compared to the BP

algorithm they have similar complexity which is approximately 20 times larger.

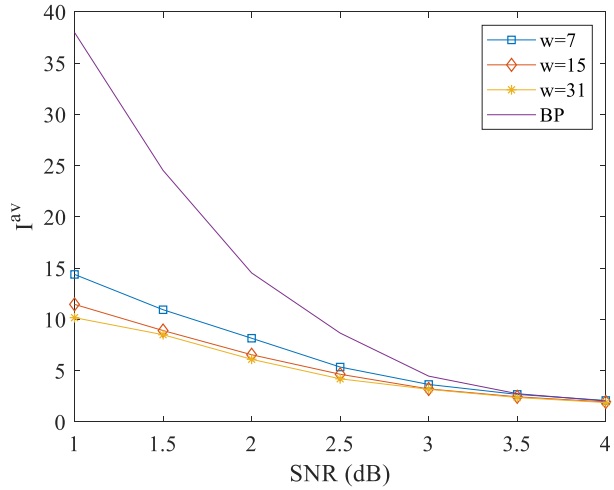


Fig. 5.13 Comparison of different bit widths w, $L_s = 20$, $L_{max} = 8$, average iteration times I^{av}

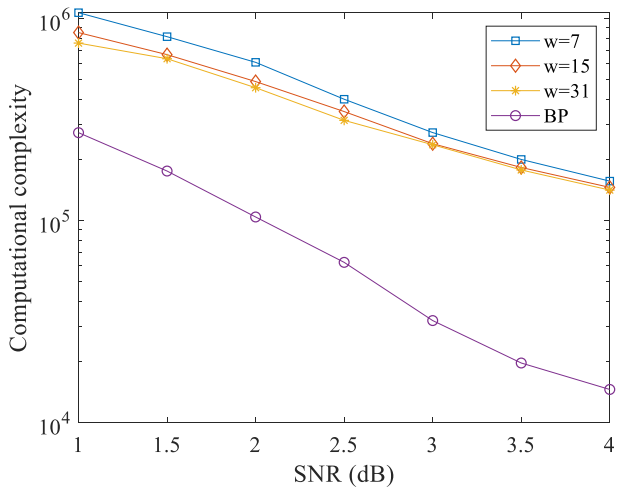


Fig. 5.14 Comparison of different bit widths w, $L_s = 20$, $L_{max} = 8$, computational complexity

Based on the simulation results, it can be supposed that with a large bit width w, the length of the non-binary quantized input sequence can be decreased without reducing the performance severely. Consequently, decoders with different combinations of sequence length L_s and sequence bit width w can be used to study their performance. Fig. 5.15-Fig. 5.18 show the performance of stochastic decoder with different combinations of the bit width w and sequence length L_s. Compared to the BP decoding method, stochastic decoding with different input sequence parameters show similar performance in the low SNR region in Fig. 5.15 and Fig. 5.16. However, in the high SNR region, the decoder with a bit width of $w = 7$ and non-binary sequence length of $L_s = 20$ shows a good tendency to have better performance. When it comes to complexity, in Fig. 5.17 and Fig. 5.18, the decoder with a bit width of $w = 31$ and non-binary input sequence length of $L_s = 5$, has the lowest computational complexity as the theory.

However, when the non-binary sequence length L_s is lower than 20, the performance of the stochastic decoder decreases too much to get the expected coding gain though the complexity is reduced. Consequently, to obtain a good performance in the high SNR region, the decoder with the quantized value of $L_{max} = 8$, non-binary sequence length of $L_s = 20$, and bit width of $w = 15$ can be chosen finally.

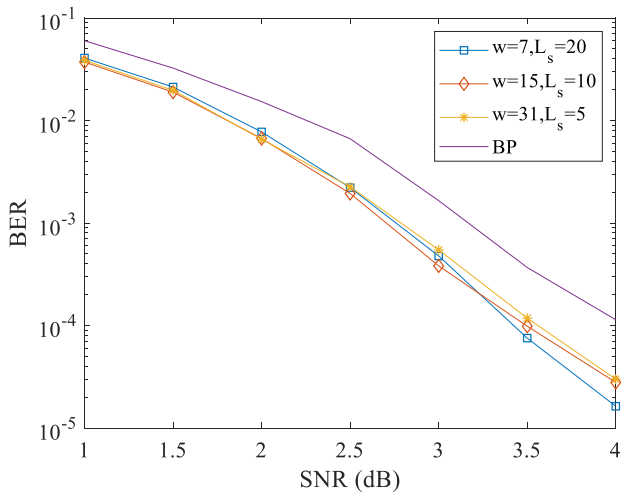


Fig. 5.15 Performance of different combinations, $L_{max} = 8$, BER

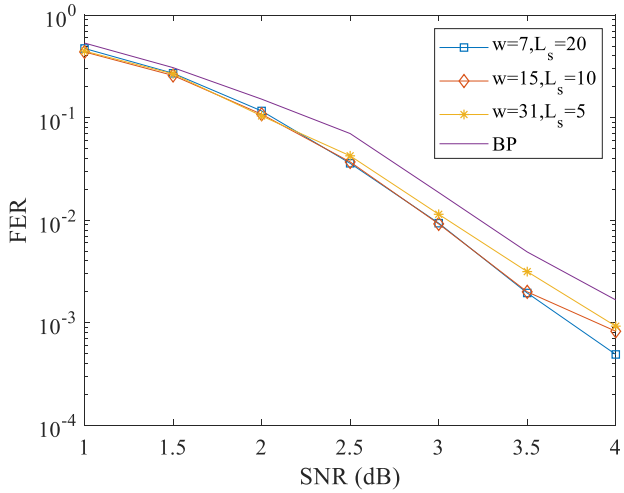


Fig. 5.16 Performance of different combinations, $L_{max} = 8$, FER

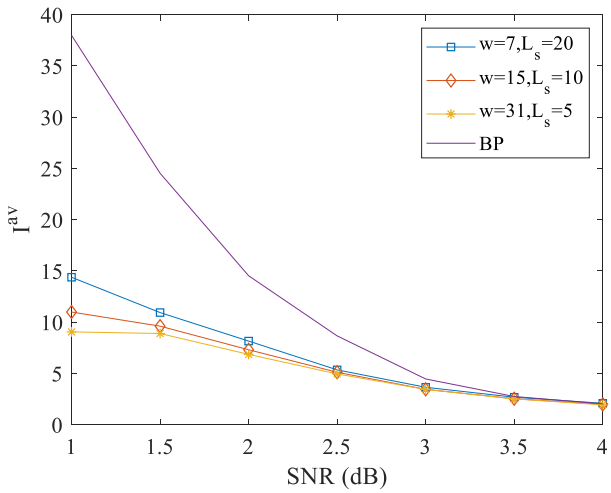


Fig. 5.17 Performance of different combinations, $L_{max} = 8$, average iteration times I^{av}

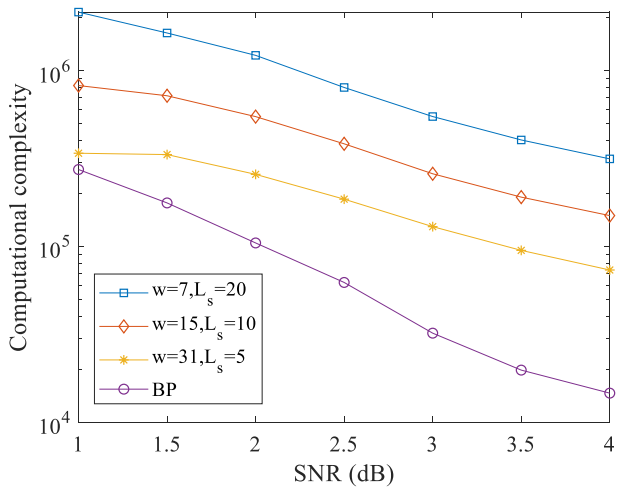


Fig. 5.18 Performance of different combinations, $L_{max} = 8$, computational complexity

6. Results

6.1. Comparison of Different Decoding Methods

The comparison of the performance of different decoding methods is shown in Fig. 6.1.

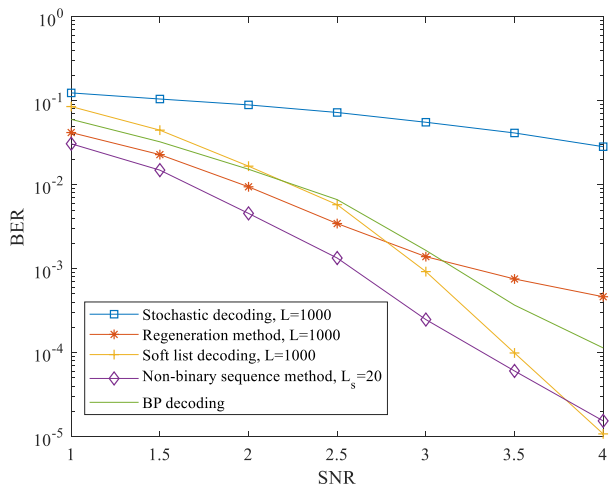


Fig. 6.1 Comparison of different decoding methods

From Fig. 6.1, among all decoding methods, the initial stochastic decoding (the blue curve) shows the worst performance. In Section 3, stochastic decoding with the regeneration method (the red curve) is put forward, resulting in improved performance. However, even with the regeneration method, stochastic decoding with the majority decision method can't achieve a better performance compared to the BP decoding algorithm. Hence, in Section 4 list decoding method has been studied to improve the performance. With the soft input sequence, the stochastic list decoding with the soft ML decision method (the yellow curve) achieves a good performance which is much lower than the one of BP decoding in high SNR region. With the consideration of reducing the complexity and further improving the performance, in Section 5, stochastic decoding with the non-binary quantized input sequence (the purple curve) is proposed. This method has a very low complexity which is about 20 times that of the BP algorithm, while stochastic list decoding has thousand times of complexity compared to the BP algorithm.

With such a low computational complexity, this decoding method can still achieve a much better performance compared to BP decoding.

6.2. Simulation Results with Other LDPC Codes

In order to verify that the designed stochastic decoder always has a good performance, some other kinds of the short LDPC codes are chosen to do the simulation with our decoder. Section 6.2.1 will focus on the influence of the 4-cycle existing in the short LDPC codes on the basis of the quasi-cyclic LDPC code which has better structure than the typical LDPC code. Section 6.2.2 will emphasize on the effect caused by the LDPC codes with large degree as (128,4,8) LDPC code. The parameters of the stochastic decoder keep the same as the ones determined in Section 5.3.2 which are quantized value of $L_{max} = 8$, non-binary sequence length of $L_s = 20$, and bit width of $w = 15$ to have the low complexity and good performance.

6.2.1. (126, 3, 6) QC LDPC Codes without 4-Cycle

Typically, LDPC code as the (126, 3, 6) LDPC code used in the previous simulations is constructed with random construction method. However, the LDPC parity-check matrix constructed in this way has the problems such as difficulties in storing the check matrix and high coding complexity. Thus, in order to avoid such problems brought with the classic LDPC codes, a quasi-cyclic (QC) LDPC code has been proposed in recent years. QC LDPC code is the most important kind of LDPC code, and it means that a codeword is still a codeword by shifting the sign bit of a fixed number of bits to the right or left. Its parity check matrix can be divided into multiple square matrices of equal size. Each square matrix is a cyclic shift matrix of the identity matrix or an all-zero matrix, which is very convenient for storage and addressing of the memory, thus greatly reducing the encoding, and decoding complexity of LDPC codes. A quasi-cyclic LDPC code with a repeating accumulation structure can realize fast coding with linear complexity [20].

Besides, unlike the (126, 3, 6) LDPC code used from Section 3 to Section 5.3, the (126, 3, 6) QC LDPC code in this section has no 4-cycle which will decrease the performance of the decoding method as mentioned in Section 2.1. The comparison of the (126, 3, 6) LDPC code and (126, 3, 6) QC LDPC code is shown in the Fig. 6.2 and Fig. 6.3.

From Fig. 6.2 and Fig. 6.3, the performance of the stochastic decoding with non-binary stochastic sequence will not be influenced by the 4-cycle in the low SNR region. However, with the SNR increasing, the performance of QC LDPC code without 4-cycle will get better than the (126, 3, 6) LDPC

code gradually which proves the bad effect of the 4-cycle. But in general, the performance of the stochastic decoding is better than the BP decoding whether with typical LDPC or with QC LDPC without 4-cycle.

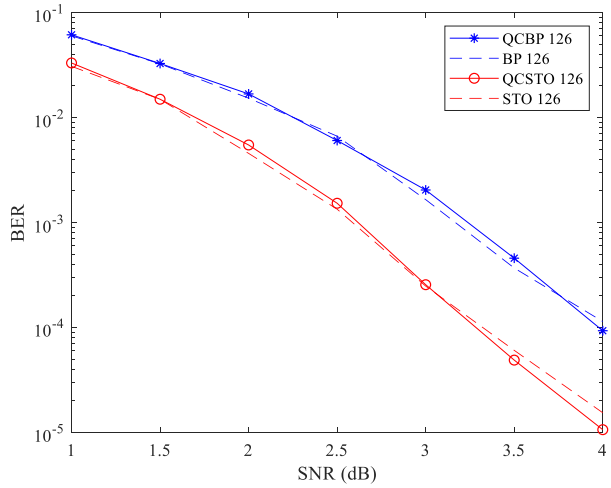


Fig. 6.2 Comparison of (126, 3, 6) LDPC code and (126, 3, 6) QC LDPC code, $L_{max} = 8$, $L_s = 20$, $w = 15$, BER

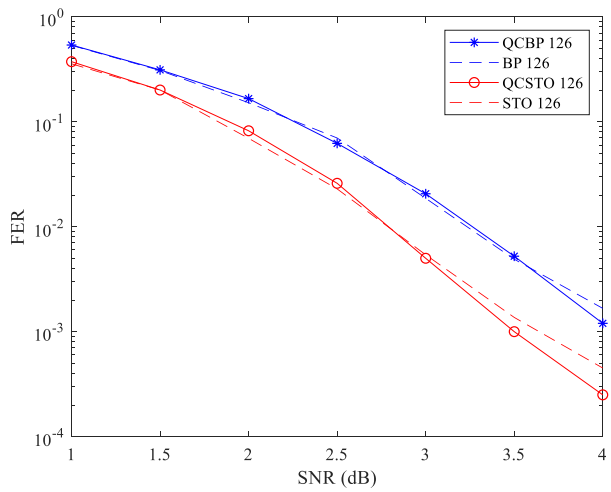


Fig. 6.3 Comparison of (126, 3, 6) LDPC code and (126, 3, 6) QC LDPC code, $L_{max} = 8$, $L_s = 20$, $w = 15$, FER

6.2.2. (128, 4, 8) QC LDPC Codes without 4-Cycle

Besides the effect of the girth distribution mentioned in Section 5.4.2 on the stochastic decoder, the degree of the nodes of LDPC codes is also a significant factor which can influence the performance of the decoding method. Fig. 6.4 and Fig. 6.5 show the comparison of the (126, 3, 6) LDPC code and (128, 4, 8) QC LDPC code.

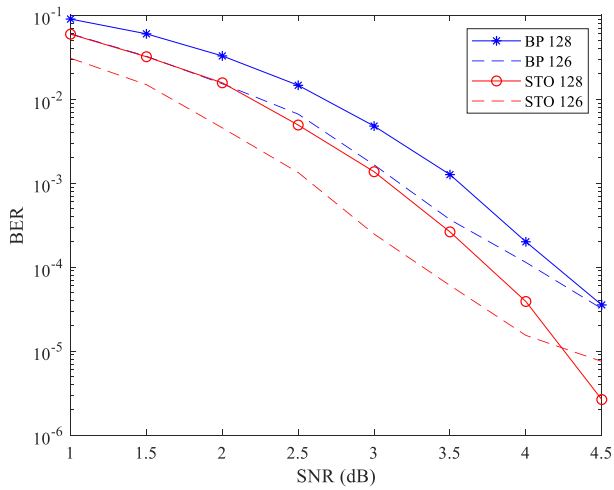


Fig. 6.4 Comparison of (126, 3, 6) LDPC code and (128, 4, 8) QC LDPC code, $L_{max} = 8$, $L_s = 20$, $w = 15$, BER

Fig. 6.4 and Fig. 6.5 clearly show the influence of large degree which decreases performance of both stochastic decoding and BP decoding in the low SNR region, since large degree causes the high density of the check matrix. However, the waterfall region of (128, 4, 8) QC LDPC code is much broader than the one of (126, 3, 6) LDPC code, which means the error rate of the high degree LDPC codes decreases more rapidly. Because with the large degree, every node can get more information from more adjacent nodes to output a higher reliable result. Thus, the BER and FER of (128, 4, 8) QC LDPC code becomes better than the performance of (126, 3, 6) LDPC code after SNR of 4dB. Comparing with the BP decoding, the stochastic decoding with non-binary quantized stochastic sequence can still keep a better performance than the BP decoding with (128, 4, 8) QC LDPC code as it does with (126, 3, 6) LDPC code.

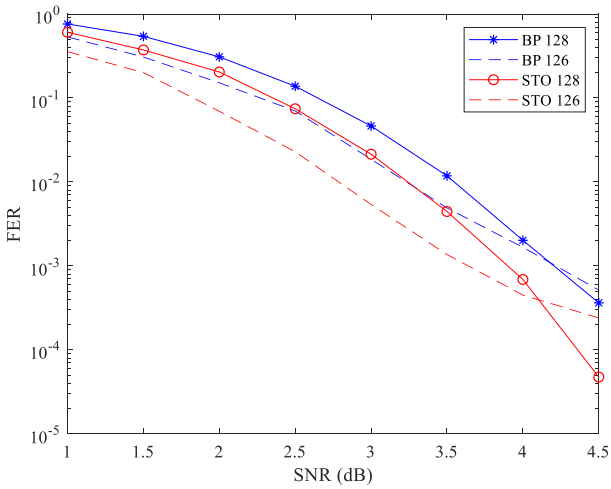


Fig. 6.5 Comparison of (126, 3, 6) LDPC code and (128, 4, 8) QC LDPC code, $L_{max} = 8$, $L_s = 20$, $w = 15$, FER

7. Conclusions

This thesis proposes several decoding methods to enhance the decoding of short LDPC codes with the stochastic sequence in the background of the URLLC scenario.

In Section 3, inspired by [5] and [7], stochastic decoding with parallel decoders is proposed although its performance turns out to be not better than BP decoding. After making research on [7], stochastic decoding is improved with the regeneration method. However, the performance of such a stochastic decoding method can only be slightly better than BP decoding in the low SNR region but is worse in the high SNR region.

Based on the bad performance of stochastic decoding introduced in Section 3, a stochastic list decoding algorithm whose decision method is ML decision on the outputs of the stochastic decoder is introduced in Section 4. Firstly, stochastic list decoding is simulated based on the stochastic decoding with regeneration, but its performance is even worse than the previous stochastic decoding method. Then, since only one codeword is too weak to decode the list with the ML decision method, an enhancement method stochastically combining every 5 resulting codewords after iterative decoding is proposed in Section 4.2. This enhancement method improves the performance of the initial stochastic list decoding, but still cannot let the stochastic list decoder be superior to BP decoding. Thus, in Section 4.3, a stochastic list decoding which transforms the binary bit input to the corresponding LLR is constructed, and it uses the BP decoding algorithm in the iterative decoding part. After the simulation, the stochastic decoding with the soft input can perform better than BP decoding, and the gap between them becomes larger with the SNR increasing.

Then, inspired by Section 4.2 and Section 4.3, stochastic decoding with non-binary sequences is proposed in Section 5. The non-binary sequence is generated with the combination of the binary input. And then, these non-binary sequences are transformed to the corresponding LLR and taken into the iterative decoding part with the BP algorithm. The best decision method after iterative decoding is found as the soft ML decision in Section 5.2.4. Finally, after several simulations on the parameters which can affect the performance, the non-binary stochastic decoder with the bit width of $w = 15$, quantized value of $L_{max} = 8$, and non-binary sequence length of $L_s = 20$ is determined to have a low decoding complexity and remarkable performance. In Section 6.2, the non-binary stochastic decoder is verified to still have a good performance with other short LDPC codes.

8. Future work

For the future work, the performance of the current designed stochastic decoder could be further verified with other kinds of short codes like BCH codes and Reed-Solomon code. Besides, the complexity of the non-binary stochastic decoder is mainly calculated from the operand in the iterative decoding which is actually a little inaccurate in our thesis. The actual complexity should be based on the implementation of the hardware, so implementation of our stochastic decoder in the hardware could be further investigated in the future. In addition, our stochastic decoder will be compared with the multiple bases BP decoder to further determine the performance.

References

[1] R. Gallager, "Low-density parity-check codes," in *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21-28, January 1962, doi: 10.1109/TIT.1962.1057683.

[2] R. Tanner, "A recursive approach to low complexity codes," in *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533-547, September 1981, doi: 10.1109/TIT.1981.1056404.

[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," in *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399-431, March 1999, doi: 10.1109/18.748992.

[4] G. Liva, L. Gaudio, T. Ninacs and T. Jerkovits, "Code design for short blocks: A survey", available at https://arxiv.org/abs/1610.00873, 2016.

[5] T. Hehn, J. B. Huber, S. Laendner and O. Milenkovic, "Multiple-Bases Belief-Propagation for Decoding of Short Block Codes," *2007 IEEE International Symposium on Information Theory*, 2007, pp. 311-315, doi: 10.1109/ISIT.2007.4557244.

[6] W. Zhou and M. Lentmaier, "Improving Short-Length LDPC Codes with a CRC and Iterative Ordered Statistic Decoding: (Invited Paper)," *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA, 2019, pp. 1-6, doi: 10.1109/CISS.2019.8693053.

[7] A. Rapley, C. Winstead, V. Gaudet, and C. Schlegel, "Stochastic iterative decoding on factor graphs," in *Proc. 3rd Int. Symp. Turbo Codes Related Topics*, Brest, France, Sep. 2003, pp. 507–510.

[8] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of Low-Density Parity-Check codes," *Electron. Lett.*, 1997.

[9] W. E. Ryan, "An introduction to LDPC codes," in *CRC Handbook for Coding and Signal Processing for Recoding Systems*, B. Vasic, Ed. Boca Raton, FL, USA: CRC, 2004.

[10] H.-A. Loeliger, "An introduction to factor graphs," in *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28-41, Jan. 2004, doi: 10.1109/MSP.2004.1267047.

[11] C. Hausl, F. Schreckenbach, I. Oikonomidis, and G. Bauch, "Iterative network and channel decoding on a tanner graph," in *43rd Annual Allerton Conference on Communication, Control and Computing*, 2005.

[12] J. Coughlan, “A Tutorial Introduction to Belief Propagation Table of Contents,” *Sixth Can. Conf. Comput. Robot Vis. CRV*, 2009.

[13] B. K. Butler and P. H. Siegel, "Error Floor Approximation for LDPC Codes in the AWGN Channel," in *IEEE Transactions on Information Theory*, vol. 60, no. 12, pp. 7416-7441, Dec. 2014, doi: 10.1109/TIT.2014.2363832.

[14] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier and Xiao-Yu Hu, "Reduced-complexity decoding of LDPC codes," in *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288-1299, Aug. 2005, doi: 10.1109/TCOMM.2005.852852.

[15] J. Zhao, F. Zarkeshvari and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density Parity-check (LDPC) codes," in *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549-554, April 2005, doi: 10.1109/TCOMM.2004.836563.

[16] Y. Jian, H. D. Pfister, K. R. Narayanan, Raghu Rao and R. Mazahreh, "Iterative hard-decision decoding of braided BCH codes for high-speed optical communication," *2013 IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, USA, 2013, pp. 2376-2381, doi: 10.1109/GLOCOM.2013.6831429.

[17] S. Sharifi Tehrani, W. J. Gross and S. Mannor, "Stochastic decoding of LDPC codes," in *IEEE Communications Letters*, vol. 10, no. 10, pp. 716-718, Oct. 2006, doi: 10.1109/LCOMM.2006.060570.

[18] P. Elias. "List decoding for noisy channels," in *Technical Report 335, Research Laboratory of Electronics*, MIT, 1957.

[19] R. Jose and A. Pe, "Analysis of hard-decision and soft-decision decoding algorithms of LDPC codes in AWGN," *2015 IEEE International Advance Computing Conference (IACC)*, pp. 430-435, doi: 10.1109/IADCC.2015.7154744, 2015.

[20] M. Esmaeili, and S. Yari, “Generalized quasi-cyclic codes: structural properties and code construction,” *Applicable Algebra in Engineering, Communication and Computing*, vol. 20, no. (2), pp.159-173, 2015.